K 1

```
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP
```

```
_$

Val
---
000
000
000
7F
7F
7F
7F
7F
7F
7F
7F
7F
```

```
**FILE**ID**UETDMPF00
```

```
UU      UU  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    MM      MM  PPPPPPPP    FFFFFFFFFF    000000      000000
UU      UU  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    MM      MM  PPPPPPPP    FFFFFFFFFF    000000      000000
UU      UU  EE              TT      DD      DD  MMMM  MMMM  PP      PP  FF          00      00  00      00
UU      UU  EE              TT      DD      DD  MMMM  MMMM  PP      PP  FF          00      00  00      00
UU      UU  EE              TT      DD      DD  MM  MM  MM  PP      PP  FF          00    0000  00    0000
UU      UU  EE              TT      DD      DD  MM  MM  MM  PP      PP  FF          00    0000  00    0000
UU      UU  EEEEEEEE        TT      DD      DD  MM      MM  PPPPPPPP    FFFFFFFF    00  00  00  00  00  00
UU      UU  EEEEEEEE        TT      DD      DD  MM      MM  PPPPPPPP    FFFFFFFF    00  00  00  00  00  00
UU      UU  EE              TT      DD      DD  MM      MM  PP          FF          0000    00  0000    00
UU      UU  EE              TT      DD      DD  MM      MM  PP          FF          0000    00  0000    00
UU      UU  EE              TT      DD      DD  MM      MM  PP          FF          00      00  00      00      ....
UU      UU  EE              TT      DD      DD  MM      MM  PP          FF          00      00  00      00      ....
UUUUUUUUUU  EEEEEEEEEE      TT      DDDDDDDD    MM      MM  PP          FF            000000      000000        ....
UUUUUUUUUU  EEEEEEEEEE      TT      DDDDDDDD    MM      MM  PP          FF            000000      000000        ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
    (2)        86       Declarations
    (3)       164       Read-Only Data
    (4)       301       Read/Write Data
    (5)       489       RMS-32 Data Structures
    (6)       543       Main Program
    (11)      852       Test the DMP/DMF
    (12)     1151       CHECKIOSB - Check IO status block
    (13)     1212       Check QIO AST Routine
    (14)     1253       Receive data AST routine
    (15)     1310       Half Minute Timer Expiration Routine
    (16)     1352       Three Minutes Timer Expiration Routine
    (18)     1387       System Service Exception Handler
    (19)     1516       RMS Error Handler
    (20)     1580       CTRL/C Handler
    (21)     1631       Error Exit
    (22)     1697       Exit Handler
```

```
0000          .TITLE UETDMPF00 VAX/VMS UETP DEVICE TEST FOR DMP 11/ DMF-32 Sync line
0000          .IDENT  'V04-001'
0000   ;
0000   ;*****************************************************************************
0000   ;*                                                                           *
0000   ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000   ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000   ;*  ALL RIGHTS RESERVED.                                                     *
0000   ;*                                                                           *
0000   ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000   ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000   ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000   ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000   ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000   ;*  TRANSFERRED.                                                             *
0000   ;*                                                                           *
0000   ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000   ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000   ;*  CORPORATION.                                                             *
0000   ;*                                                                           *
0000   ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000   ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000   ;*                                                                           *
0000   ;*                                                                           *
0000   ;*****************************************************************************
0000   ;
0000
0000   ;++
0000   ; FACILITY:
0000   ;       This module will be distributed with VAX/VMS under the [SYSTEST]
0000   ;       account.
0000   ;
0000   ; ABSTRACT:
0000   ;       This is the test program for DMP 11 / DMF 32 sync line device test
0000   ;
0000   ; ENVIRONMENT:
0000   ;       This program will run in user access mode, with AST enabled except
0000   ;       during error processing. This program requires the following
0000   ;       privileges and  quotas:  none.
0000   ;
0000   ;--
0000   ;
0000   ; AUTHOR: Paul Jenq,    CREATION DATE: Sep, 1981
0000   ;
0000   ; MODIFIED BY:
0000   ;
0000   ;       V04-001 RNH0009         Richard N. Holstein,    07-Sep-1984
0000   ;               Remove entirely the forced error for too big a buffer, since
0000   ;               either SS$_BADPARAM or SS$_EXQUOTA could be returned depending
0000   ;               on SYSGEN parameters.
0000   ;
0000   ;       V03-010 RNH0008         Richard N. Holstein,    07-Apr-1984
0000   ;               Adapt to driver fix which allocated write buffers dynamically -
0000   ;               we can't force an SS$_BADPARAM unless we exceed absolute max.
0000   ;
0000   ;       V03-009 RNH0007         Richard N. Holstein,    15-Feb-1984
0000   ;               Take advantage of the new UETP message codes.  Fix SSERROR
```

```
0000   ;                interaction with RMS_ERROR.
0000   ;
0000   ;        V03-008 RNH0006          Richard N. Holstein,    19-Dec-1983
0000   ;                Give correct sentinels to Test Controller.
0000   ;
0000   ;        V03-007 RNH0005          Richard N. Holstein,    11-Nov-1983
0000   ;                Use decimal conversion routine for unit numbers.
0000   ;
0000   ;        V03-006 RNH0004          Richard N. Holstein,    29-Jun-1983
0000   ;                Rework error messages and error processing.
0000   ;
0000   ;        V03-005 RNH0003          Richard N. Holstein,    11-Mar-1983
0000   ;                Don't signal ending message in EXIT_HANDLER.
0000   ;
0000   ;        V03-004 RNH0002          Richard N. Holstein,    01-Mar-1983
0000   ;                Fix ERROR_COUNT bug.
0000   ;
0000   ;        V03-003 LDJ0002          Larry D. Jones,         10-Feb-1983
0000   ;                Allow for longer device names.
0000   ;
0000   ;        V03-002 LDJ0001          Larry D. Jones,         06-Nov-1982
0000   ;                Fixed a loop mode assign channel bug.
0000   ;
0000   ;        V03-001 RNH0001          Richard N. Holstein,    15-Oct-1982
0000   ;                Miscellaneous fixes listed in the V3B UETP Workplan.
0000   ;
0000   ;**
```

```
          0000          .SBTTL  Declarations
          0000  ;
          0000  ; INCLUDE FILES:
          0000  ;
          0000  ;       SYS$LIBRARY:LIB.MLB      for general definitions
          0000  ;       SHRLIB$:UETP.MLB         for UETP definitions
          0000
          0000  ;
          0000  ; MACROS:
          0000  ;
          0000          $CHFDEF                                 ; Condition handler frame definitions
          0000          $DEVDEF                                 ; Device definitions
          0000          $DIBDEF                                 ; Device Information Block
          0000          $DVIDEF                                 ; $GETDVI ITMLST item codes
          0000          $SHRDEF                                 ; Shared messages
          0000          $SSDEF                                  ; System Service status codes
          0000          $STSDEF                                 ; Status return
          0000          $UETUNTDEF                              ; UETP unit block offset definitions
          0000          $UETPDEF                                ; UETP
          0000          $XMDEF                                  ; XMDRIVER symbols
          0000          $NMADEF                                 ; Network management definition
          0000  ;
          0000  ; EQUATED SYMBOLS:
          0000  ;
          0000  ;   Facility number definitions:
00000001  0000          RMS$_FACILITY = 1
          0000
          0000  ;   SHR message definitions:
00740000  0000          UETP = UETP$_FACILITY@STS$V_FAC_NO ; Define the UETP facility code
007410E0  0000          UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
00741038  0000          UETP$_BEGIND = UETP!SHR$_BEGIND
00741080  0000          UETP$_ENDEDD = UETP!SHR$_ENDEDD
00741098  0000          UETP$_OPENIN = UETP!SHR$_OPENIN
00741130  0000          UETP$_TEXT   = UETP!SHR$_TEXT
          0000
          0000  ;   Internal flag bits...:
00000001  0000          TEST_OVERV   = 1                        ; Set when test is over
00000002  0000          SAFE_TO_UPDV = 2                        ; Set if it's safe to update UETINIDEV
00000003  0000          BEGIN_MSGV   = 3                        ; Set if "BEGIN" msg has been printed
00000004  0000          MODE_IS_ONEV = 4                        ; Set when the MODE is ONE
00000005  0000          FLAG_SHUTDNV = 5                        ; Set to indicate device should be
          0000                                                  ; shutdown if errors occur
          0000  ;   ...and corresponding masks:
00000002  0000          TEST_OVERM   = 1@TEST_OVERV
00000004  0000          SAFE_TO_UPDM = 1@SAFE_TO_UPDV
00000008  0000          BEGIN_MSGM   = 1@BEGIN_MSGV
00000010  0000          MODE_IS_ONEM = 1@MODE_IS_ONEV
00000020  0000          FLAG_SHUTDNM = 1@FLAG_SHUTDNV
          0000
          0000  ;   Miscellany:
00000020  0000          LC_BITM        = ^X20                   ; Mask to convert lower case to upper
00000028  0000          REC_SIZE       = 40                     ; UETINIDEV.DAT record size
000000FA  0000          TEXT_BUFFER    = 250                    ; Internal text buffer size
00000004  0000          EFN2           = 4                      ; EFN used for three minute timer
00000003  0000          SS_SYNCH_EFN   = 3                      ; Synch miscellaneous system services
0000000F  0000          MAX_PROC_NAME  = 15                     ; Longest possible process name
0000000A  0000          MAX_DEV_DESIG  = 10                     ; Longest possible controller name
```

```
00000005  0000   143          MAX_UNIT_DESIG= 5                    ; Longest possible unit number
00000200  0000   144          MAX_MSG_LEN    = 512                 ; maximum message length
00000001  0000   145          TIME_ID_1      = 1                   ; Timer id to prevent hung
00000003  0000   146          RW_TIME_ID     = 3                   ; Timer to prevent hung when Read/write
00000010  0000   147          LIMIT          = 16                  ; Loop count for each message length
00000008  0000   148          RECV_EFN       = 8
00000005  0000   149          XMIT_EFN       = 5                   ; EFN for QIO write
00000064  0000   150          PRM            = 100                 ; AST parameter for test
00000000  0000   151          DEVDEP_SIZE    = 0                   ; Size of device dependent part of UETUNT
00000000  0000   152          WRITE_SIZE     = 0                   ; Size of device write buffer
00000000  0000   153          READ_SIZE      = 0                   ; Size of device read buffer
          0000   154
          0000   155          PAGES = <<UETUNT$C_INDSIZ+-          ; Add together all of the pieces...
          0000   156                    DEVDEP_SIZE+-              ; ...which make up a UETP unit block...
          0000   157                    WRITE_SIZE+-               ; ...to give to the $EXPREG service below
          0000   158                    READ_SIZE+-
00000001  0000   159                    511>/512>
          0000   160
0000001B  0000   161          ESC = ^X1B                           ; ESC character
00000004  0000   162          RECVPOOL_SIZ = 4                     ; Number of preallocated message block
```

```
                                        0000                .SBTTL  Read-Only Data
                                    00000000                .PSECT  RODATA,NOEXE,NOWRT,PAGE
                                        0000
                                        0000      ACNT_NAME:                                  ; Process name on exit
53 45 54 53 59 53 00000008'010E0000'    0000                .ASCID  /SYSTEST/
                                  54    000E
                                        000F
                                        000F      TEST_NAME:                                  ; This test name
50 4D 44 54 45 55 00000017'010E0000'    000F                .ASCID  /UETDMPF00/
                            30 30 46    001D
                                        0020
                                        0020      SUPDEV_GBLSEC:                              ; How we access UETSUPDEV.DAT
50 55 53 54 45 55 00000028'010E0000'    0020                .ASCID  /UETSUPDEV/
                            56 45 44    002E
                                        0031
                                        0031      CONTROLLER:                                 ; Logical name of controller
41 4E 4C 52 54 43 00000039'010E0000'    0031                .ASCID  /CTRLNAME/
                               45 4D    003F
                                        0041
                                        0041      MODE:                                       ; Run mode logical name
      45 44 4F 4D 00000049'010E0000'    0041                .ASCID  /MODE/
                                        004D
                                        004D      NO_RMS_AST_TABLE:                           ; List of errors for which...
                           00000000'    004D                .LONG   RMS$_BLN                  ; ...RMS cannot deliver an AST...
                           00000000'    0051                .LONG   RMS$_BUSY                 ; ...even if one has an ERR= arg
                           00000000'    0055                .LONG   RMS$_CDA                  ; Note that we can search table...
                           00000000'    0059                .LONG   RMS$_FAB                  ; ...via MATCHC since <31:16>...
                           00000000'    005D                .LONG   RMS$_RAB                  ; ...pattern can't be in <15:0>
                           00000014     0061      NRAT_LENGTH = .-NO_RMS_AST_TABLE
                                        0061
                                        0061      SYS$INPUT:                                  ; Name of device from which...
4E 49 24 53 59 53 00000069'010E0000'    0061                .ASCID  /SYS$INPUT/               ; ...the test can be aborted
                            54 55 50    006F
                                        0072
                                        0072      INPUT_ITMLST:                               ; $GETDVI arg list for SYS$INPUT
                           0020 0040    0072                .WORD   64,DVI$_DEVNAM            ; We need the equivalence name
                 0000000C'00000014'     0076                .LONG   BUFFER,BUFFER_PTR
                           00000000     007E                .LONG   0                         ; Terminate the list
                                        0082
                                        0082      CS1:                                        ; Device class and type control string
21 20 42 58 32 21 0000008A'010E0000'    0082                .ASCID  /!2XB !2XB /
                         20 42 58 32    0090
                                        0094
                                        0094      CS3:                                        ; Device class-only control string
2A 20 42 58 32 21 0000009C'010E0000'    0094                .ASCID  /!2XB **/
                                  2A    00A2
                                        00A3
                                        00A3      CNTRLCMSG:
65 74 72 6F 62 41 000000AB'010E0000'    00A3                .ASCID  \Aborted via a user CTRL/C\
72 65 73 75 20 61 20 61 69 76 20 64     00B1
            43 2F 4C 52 54 43 20        00BD
                                        00C4
                                        00C4      NO_CTRLNAME:
6E 6F 63 20 6F 4E 000000CC'010E0000'    00C4                .ASCID  /No controller specified./
63 65 70 73 20 72 65 6C 6C 6F 72 74     00D2
               2E 64 65 69 66 69        00DE
                                        00E4
```

```
                                          00E4   210 DEAD_CTRLNAME:
20 74 27 6E 61 43 000000EC'010E0000'  00E4   211         .ASCID  /Can't test controller !AS, marked as unusable in UETINIDEV.DAT./
6C 6F 72 74 6E 6F 63 20 74 73 65 74  00F2
72 61 6D 20 2C 53 41 21 20 72 65 6C  00FE
61 73 75 6E 75 20 73 61 20 64 65 6B  010A
4E 49 54 45 55 20 6E 69 20 65 6C 62  0116
         2E 54 41 44 2E 56 45 44 49  0122
                                          012B   212
                                          012B   213 NOUNIT_SELECTED:
69 6E 75 20 6F 4E 00000133'010E0000'  012B   214         .ASCID  /No units selected for testing./
20 64 65 74 63 65 6C 65 73 20 73 74  0139
2E 67 6E 69 74 73 65 74 20 72 6F 66  0145
                                          0151   215
                                          0151   216 ILLEGAL_REC:
61 67 65 6C 6C 49 00000159'010E0000'  0151   217         .ASCID  /Illegal record format in file UETINIDEV.DAT!/
72 6F 66 20 64 72 6F 63 65 72 20 6C  015F
20 65 6C 69 66 20 6E 69 20 74 61 6D  016B
41 44 2E 56 45 44 49 4E 49 54 45 55  0177
                                21 54  0183
                                          0185   218
                                          0185   219 PASS_MSG:
66 6F 20 64 6E 45 0000018D'010E0000'  0185   220         .ASCID  /End of pass !UL with !UL iterations at !%D./
69 77 20 4C 55 21 20 73 73 61 70 20  0193
61 72 65 74 69 20 4C 55 21 20 68 74  019F
44 25 21 20 74 61 20 73 6E 6F 69 74  01AB
                                   2E  01B7
                                          01B8   221
                                          01B8   222 INIDEV_UPDERR:                          ; Error during exit handler
20 72 6F 72 72 45 000001C0'010E0000'  01B8   223         .ASCID  /Error updating UETINIDEV.DAT./
54 45 55 20 67 6E 69 74 61 64 70 75  01C6
   2E 54 41 44 2E 56 45 44 49 4E 49  01D2
                                          01DD   224
                                          01DD   225 THREEMIN:                               ; 3 minute delta time
                   FFFFFFFF 94B62E00  01DD   226         .LONG   -10*1000*1000*180,-1
                                          01E5   227
                                          01E5   228 HALFMIN:                                ; 30 seconds delta time
                   FFFFFFFF EE1E5D00  01E5   229         .LONG   -10*1000*1000*30,-1
                                          01ED   230
                                          01ED   231 UNIT_DESC:                              ; Descriptor used to convert unit #
                            00000005  01ED   232         .LONG   5
                           0000001A'  01F1   233         .ADDRESS BUFFER+6
                                          01F5   234
                                          01F5   235 CONT_DESC:                              ; Descriptor used to convert controller...
                           0000 0028  01F5   236         .WORD   REC_SIZE,0                  ; ...from lowercase to uppercase
                           00000014'  01F9   237         .ADDRESS BUFFER
                                          01FD   238
                                          01FD   239 FILE:                                   ; Fills in RMS_ERR_STRING
      65 6C 69 66 00000205'010E0000'  01FD   240         .ASCID  /file/
                                          0209   241
                                          0209   242 RECORD:                                 ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 00000211'010E0000'  0209   243         .ASCID  /record/
                                          0217   244
                                          0217   245 RMS_ERR_STRING:                         ; Announces an RMS error
41 21 20 53 4D 52 0000021F'010E0000'  0217   246         .ASCID  /RMS !AS error in file !AD/
66 20 6E 69 20 72 6F 72 72 65 20 53  0225
         44 41 21 20 65 6C 69  0231
                                          0238   247
                                          0238    248 PROMPT:
64 20 72 65 6C 6C 6F 72 74 6E 6F 43      0238    249         .ASCII  /Controller designation?: /
3A 3F 6E 6F 69 74 61 6E 67 69 73 65      0244
                                    20    0250
                              00000019    0251    250         PMTSIZ = .-PROMPT
                                          0251    251
                                          0251    252 START_TO_MSG:
75 6F 65 6D 69 54 00000259'010E0000'     0251    253         .ASCID  /Timeout trying to start !AS./
20 6F 74 20 67 6E 69 79 72 74 20 74      025F
      2E 53 41 21 20 74 72 61 74 73      026B
                                          0275    254
                                          0275    255 RW_TO_MSG:
75 6F 65 6D 69 54 0000027D'010E0000'     0275    256         .ASCID  /Timeout while reading or writing !AS./
64 61 65 72 20 65 6C 69 68 77 20 74      0283
69 74 69 72 77 20 72 6F 20 67 6E 69      028F
               2E 53 41 21 20 67 6E      029B
                                          02A2    257
                                          02A2    258 START_CONT_PRM:
20 72 6F 72 72 45 000002AA'010E0000'     02A2    259         .ASCID  /Error starting up !AS as a controller./
20 70 75 20 67 6E 69 74 72 61 74 73      02B0
6E 6F 63 20 61 20 73 61 20 53 41 21      02BC
         2E 72 65 6C 6C 6F 72 74         02C8
                                          02D0    260
                                          02D0    261 START_TRIB_PRM:
20 72 6F 72 72 45 000002D8'010E0000'     02D0    262         .ASCID  /Error starting up !AS as a tributary./
20 70 75 20 67 6E 69 74 72 61 74 73      02DE
69 72 74 20 61 20 73 61 20 53 41 21      02EA
            2E 79 72 61 74 75 62         02F6
                                          02FD    263
                                          02FD    264 WRITE_PRM:
20 72 6F 72 72 45 00000305'010E0000'     02FD    265         .ASCID  /Error writing to !AS./
21 20 6F 74 20 67 6E 69 74 69 72 77      030B
                              2E 53 41   0317
                                          031A    266
                                          031A    267 READ_PRM:
20 72 6F 72 72 45 00000322'010E0000'     031A    268         .ASCID  /Error reading from !AS./
6D 6F 72 66 20 67 6E 69 64 61 65 72      0328
                     2E 53 41 21 20      0334
                                          0339    269
                                          0339    270 SENSE_PRM:
20 72 6F 72 72 45 00000341'010E0000'     0339    271         .ASCID  /Error sensing characteristics of !AS./
72 61 68 63 20 67 6E 69 73 6E 65 73      0347
20 73 63 69 74 73 69 72 65 74 63 61      0353
            2E 53 41 21 20 66 6F         035F
                                          0366    272
                                          0366    273 SET_PRM:
20 72 6F 72 72 45 0000036E'010E0000'     0366    274         .ASCID  /Error setting characteristics of !AS./
72 61 68 63 20 67 6E 69 74 74 65 73      0374
20 73 63 69 74 73 69 72 65 74 63 61      0380
            2E 53 41 21 20 66 6F         038C
                                          0393    275
                                          0393    276 DMF_IOSB_DUMP:
74 73 20 4F 2F 49 0000039B'010E0000'     0393    277         .ASCID  \I/O status block completion status: !XW, transfer size: !XW,\-
63 20 6B 63 6F 6C 62 20 73 75 74 61      03A1
74 73 20 6E 6F 69 74 65 6C 70 6D 6F      03AD
74 20 2C 57 58 21 20 3A 73 75 74 61      03B9
65 7A 69 73 20 72 65 66 73 6E 61 72      03C5
```

```
                  2C 57 58 21 20 3A  03D1
65 74 63 61 72 61 68 63 5F 21 2F 21  03D7   278                  \!/!_characteristics: !XB, status: !XB, error summary: !XB.\
42 58 21 20 3A 73 63 69 74 73 69 72  03E3
58 21 20 3A 73 75 74 61 74 73 20 2C  03EF
6D 75 73 20 72 6F 72 72 65 20 2C 42  03FB
      2E 42 58 21 20 3A 79 72 61 6D  0407
                                     0411   279
                                     0411   280 DMP_IOSB_DUMP:
74 73 20 4F 2F 49 00000419'010E0000' 0411   281         .ASCID  \I/O status block completion status: !XW, transfer size: !XW,\-
63 20 6B 63 6F 6C 62 20 73 75 74 61  041F
74 73 20 6E 6F 69 74 65 6C 70 6D 6F  042B
74 20 2C 57 58 21 20 3A 73 75 74 61  0437
65 7A 69 73 20 72 65 66 73 6E 61 72  0443
                  2C 57 58 21 20 3A  044F
65 74 63 61 72 61 68 63 5F 21 2F 21  0455   282                  \!/!_characteristics: !XB, status: !XB, error summary: !XB,\-
42 58 21 20 3A 73 63 69 74 73 69 72  0461
58 21 20 3A 73 75 74 61 74 73 20 2C  046D
6D 75 73 20 72 6F 72 72 65 20 2C 42  0479
      2C 42 58 21 20 3A 79 72 61 6D  0485
75 6E 20 6C 61 74 6F 74 5F 21 2F 21  048F   283                  \!/!_total number of errors: !XB.\
6F 72 72 65 20 66 6F 20 72 65 62 6D  049B
            2E 42 58 21 20 3A 73 72  04A7
                                     04AF   284
                                     04AF   285 COMP_STATUS_MSG:
72 75 6C 69 61 46 000004B7'010E0000' 04AF   286         .ASCID  /Failure during forced error tests,/<13><10><9>/expected: "/
72 6F 66 20 67 6E 69 72 75 64 20 65  04BD
65 74 20 72 6F 72 72 65 20 64 65 63  04C9
63 65 70 78 65 09 0A 0D 2C 73 74 73  04D5
                  22 20 3A 64 65 74  04E1
                                     04E7   287
                                     04E7   288 RECEIVED_MSG:
72 09 0A 0D 2C 22 000004EF'010E0J00' 04E7   289         .ASCID  /",/<13><10><9>/received: "/
      22 20 3A 64 65 76 69 65 63 65  04F5
                                     04FF   290
                                     04FF   291 RECV_ERR_MSG:
76 69 65 63 65 52 00000507'010E0000' 04FF   292         .ASCID  /Received message error, good data is !XB, bad data is !XB /
65 20 65 67 61 73 73 65 6D 20 64 65  050D
64 20 64 6F 6F 67 20 2C 72 6F 72 72  0519
20 2C 42 58 21 20 73 69 20 61 74 61  0525
20 73 69 20 61 74 61 64 20 64 61 62  0531
                        20 42 58 21  053D
                                     0541   293
                                     0541   294 SENSE_ERRMSG:
20 72 6F 72 72 45 00000549'010E0000' 0541   295         .ASCID  \Error in sense mode test, extended characteristic parameter\-
64 6F 6D 20 65 73 6E 65 73 20 6E 69  054F
65 74 78 65 20 2C 74 73 65 74 20 65  055B
74 63 61 72 61 68 63 20 64 65 64 6E  0567
61 72 61 70 20 63 69 74 73 69 72 65  0573
                     72 65 74 65 6D  057F
68 74 69 77 20 57 58 21 5F 21 2F 21  0584   296                  \!/!_!XW with value !XL not matched by any of those returned.\
6E 20 4C 58 21 20 65 75 6C 61 76 20  0590
62 20 64 65 68 63 74 61 6D 20 74 6F  059C
6F 68 74 20 66 6F 20 79 6E 61 20 79  05A8
2E 64 65 6E 72 75 74 65 72 20 65 73  05B4
                                     05C0   297
                                     05C0   298 ERRTEST_MSG:
20 72 6F 72 72 45 000005C8'010E0000' 05C0   299         .ASCID  /Error in error test /
```

```
73 65 74 20 72 6F 72 72 65 20 6E 69  05CE
                              20 74  05DA
```

```
                          05DC            .SBTTL  Read/Write Data
                      00000000            .PSECT  RWDATA,WRT,NOEXE,PAGE
                          0000
                          0000  TTCHAN:                                          ; Channel associated with ctrl. term.
                    0000  0000            .WORD   0
                          0002
                          0002  FLAG:                                            ; Miscellaneous flag bits
                    0000  0002            .WORD   0                              ; (See Equated Symbols for definitions)
                          0004
                          0004  FAO_BUF:                                         ; FAO output string descriptor
               0000 00FA  0004            .WORD   TEXT_BUFFER,0
               00000014'  0008            .ADDRESS BUFFER
                          000C
                          000C  BUFFER_PTR:                                      ; Fake .ASCID buffer for misc. strings
               0000 00FA  000C            .WORD   TEXT_BUFFER,0                  ; A word for length, a word for desc.
               00000014'  0010            .ADDRESS BUFFER
                          0014
                          0014  BUFFER:                                          ; FAO output and other misc. buffer
                0000010E  0014            .BLKB   TEXT_BUFFER
                          010E
                          010E  ALT_FAO_BUF:                                     ; FAO output string descriptor...
               0000 00FA  010E            .WORD   TEXT_BUFFER,0                  ; ...during ASTs
               0000011E'  0112            .ADDRESS ALT_BUFFER
                          0116
                          0116  ALT_BUFFER_PTR:                                  ; Fake .ASCID buffer for misc. strings
               0000 00FA  0116            .WORD   TEXT_BUFFER,0                  ; A word for length, a word for desc.
               0000011E'  011A            .ADDRESS ALT_BUFFER                    ; Used during ASTs
                          011E
                          011E  ALT_BUFFER:                                      ; FAO output and other misc. buffer...
                00000218  011E            .BLKB   TEXT_BUFFER                    ; ...during ASTs
                          0218
                          0218  DEVDSC:                                          ; Device name descriptor
               0000 000A  0218            .WORD   MAX_DEV_DESIG,0
               00000237'  021C            .ADDRESS DEV_NAME
                          0220
                          0220  PROCESS_NAME:                                    ; Process name
46 50 4D 44 00000228'010E0000'  0220      .ASCID  /DMPF/
                          022C
                0000000B  022C            PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
                00000237  022C            .BLKB   PROCESS_NAME_FREE
                          0237
                          0237  DEV_NAME:                                        ; Device name buffer
                00000246  0237            .BLKB   MAX_DEV_DESIG+MAX_UNIT_DESIG
                0000000F  0246            NAME_LEN = .-DEV_NAME
                          0246
                          0246  DIB:                                             ; Device Information Block
               0000 0074  0246            .WORD   DIB$K_LENGTH,0
               0000024E'  024A            .ADDRESS DIBBUF
                          024E  DIBBUF:
                000002C2  024E            .BLKB   DIB$K_LENGTH
                          02C2
                          02C2  ERROR_COUNT:                                     ; Cumulative error count at runtime
                00000000  02C2            .LONG   0
                          02C6
                          02C6  STATUS:                                          ; Status value on program exit
                00000000  02C6            .LONG   0
                          02CA
```

```
                   02CA  QUAD_STATUS:                          ; IO status block for misc sys. svcs.
 00000000 00000000 02CA          .QUAD   0
                   02D2
                   02D2  INADDRESS:                            ; $CRMPSC address storage
 00000000 00000000 02D2          .LONG   0,0
                   02DA  OUTADDRESS:
 00000000 00000000 02DA          .LONG   0,0
                   02E2
                   02E2  UNIT_NUMBER:                          ; Current dev unit number
              0000 02E2          .WORD   0
                   02E4
                   02E4  DEVNAM_LEN:                           ; Current device name length
              0000 02E4          .WORD   0
                   02E6
                   02E6  ITERATION:                            ; # of times all tests were executed
          00000000 02E6          .LONG   0
                   02EA
                   02EA  PASS:                                 ; Pass count
          00000000 02EA          .LONG   0
                   02EE
                   02EE  MSG_BLOCK:                            ; Auxiliary $GETMSG info
          000002F2 02EE          .BLKB   4
                   02F2
                   02F2  EXIT_DESC:                            ; Exit handler descriptor
          00000000 02F2          .LONG   0
         00000C26' 02F6          .ADDRESS EXIT_HANDLER
          00000001 02FA          .LONG   1
         000002C6' 02FE          .ADDRESS STATUS
                   0302
                   0302  ARG_COUNT:                            ; Argument counter used by ERROR_EXIT
          00000000 0302          .LONG   0
                   0306
                   0306  XD_CHAN:                              ; DMP/F circuit channel
              0000 0306          .WORD   0
                   0308
                   0308  BUF_LEN:                              ; Length of primary chars
              0000 0308          .WORD   0
                   030A
                   030A  BUF_DESC:                             ; Get channnel char buffer descriptor
          00000074 030A          .LONG   DIB$K_LENGTH
         00000312' 030E          .LONG   CHAN_BUF
                   0312
                   0312  CHAN_BUF:                             ; Channel char buffer
          00000386 0312          .BLKB   DIB$K_LENGTH
                   0386
                   0386  P1BUF:                                ; P1 Device char buffer
 00000000 00000000 0386          .QUAD   0
                   038E
                   038E  TR_P1BUF:                             ; p1 buufer for trib
 00000000 00000000 038E          .QUAD   0
                   0396
                   0396  P2BUF_DESC:                           ; P2 extended char buffer
         0000000C' 0396          .LONG   P2BUF_LEN
         0000039E' 039A          .ADDRESS P2BUF
                   039E
                   039E  P2BUF:                                ; P2 extended buffer
              0458 039E          .WORD   NMA$C_PCLI_PRO        ; Protocol mode
```

```
                  00000000  03A0          .LONG   NMA$C_LINPR_POI          ; DDCMP proint-to-point mode
                            03A4
                      0456  03A4          .WORD   NMA$C_PCLI_CON           ; Controller mode
                  00000001  03A6          .LONG   NMA$C_LINCN_LOO          ; Loopback mode
                            03AA
                  0000000C  03AA  P2BUF_LEN = .-P2BUF
                            03AA
                            03AA  TR_P2BUF_DESC:                           ; P2 extended char buffer for trib
                 00000006'  03AA          .LONG   TR_P2BUF_LEN
                 000003B2'  03AE          .ADDRESS TR_P2BUF
                            03B2
                            03B2  TR_P2BUF:                                ; P2 extended buffer for trib
                      0474  03B2          .WORD   NMA$C_PCCI_TRI           ; tributary address
                  00000001  03B4          .LONG   1                        ; Address
                            03B8
                  00000006  03B8  TR_P2BUF_LEN = .-TR_P2BUF
                            03B8
                            03B8  SENSE_P1BUF:                             ; P1 buffer for sense mode test
         00000000 00000000  03B8          .QUAD   0
                            03C0
                            03C0  SENSE_P2DESC:                            ; P2 buffer descrip for sense mode test
                 00000090'  03C0          .LONG   SENSE_P2LEN
                 000003C8'  03C4          .ADDRESS SENSE_P2BUF
                            03C8
                            03C8  SENSE_P2BUF:                             ; P2 buffer for sense mode test
                  00000458  03C8          .BLKW   <3*24>                   ; 8 quad quad words for dev information
                  00000090  0458  SENSE_P2LEN = .-SENSE_P2BUF              ; P2 buffer length
                            0458
                            0458  ERRTST_P2DESC:                           ; P2 desc for error test
                 00000008'  0458          .LONG   ERRTST_P2LEN
                 00000460'  045C          .ADDRESS ERRTST_P2BUF
                            0460
                            0460  ERRTST_P2BUF:                            ; P2 buffer for error test
                  00000468  0460          .BLKQ   1
                  00000008  0468  ERRTST_P2LEN = .-ERRTST_P2BUF
                            0468
                            0468  ERRCOUNT_DESC:                           ; Error counter buffer descrip
                 00000200'  0468          .LONG   ERRCNT_LEN
                 00000470'  046C          .ADDRESS ERRCNT_BUF
                            0470
                            0470  ERRCNT_BUF:                              ; Buffer for error counters
                  00000670  0470          .BLKQ   64
                  00000200  0670  ERRCNT_LEN = .-ERRCNT_BUF                ; Buffer length
                            0670
                            0670  XD_IOSB:                                 ; QIO IO status block for transmit
                  00000678  0670          .BLKQ   1
                            0678
                            0678  RCV_IOSB:                                ; QIO Io status block for receive
                  00000680  0678          .BLKQ   1
                            0680
                            0680  XMIT_BUF:                                ; Transmit buffer
                  00000880  0680          .BLKB   MAX_MSG_LEN
                            0880
                            0880  RECV_BUF:                                ; Receive buffer
                  00000A80  0880          .BLKB   MAX_MSG_LEN
                            0A80
                            0A80  BAD_DATA:                                ; Received wrong data
                  00  0A80    472          .BYTE   0
                      0A81    473
                      0A81    474 GOOD_DATA:                          ; Data sent (good)
                  00  0A81    475          .BYTE   0
                      0A82    476
                      0A82    477
                      0A82    478 ;
                      0A82    479 ; Head of self-relative UETP unit block queue.
                      0A82    480 ;
                      0A82    481          .ALIGN QUAD
                      0A88    482
                      0A88    483 UNIT_LIST:                          ; Head of unit block circular list
   00000000 00000000  0A88    484          .QUAD   0
                      0A90    485
                      0A90    486 NEW_NODE:                           ; Newly acquired node address
   00000000 00000000  0A90    487          .QUAD   0
```

UET
V04-

```
                0A98          .SBTTL  RMS-32 Data Structures
                0A98          .ALIGN  LONG
                0A98
                0A98  SYSIN_FAB:                                  ; Allocate FAB for SYS$INPUT
                0A98          $FAB-
                0A98          FNM = <SYS$INPUT>
                0AE8
                0AE8  SYSIN_RAB:                                  ; Allocate RAB for SYS$INPUT
                0AE8          $RAB-
                0AE8          FAB = SYSIN_FAB,-
                0AE8          ROP = PMT,-
                0AE8          PBF = PROMPT,-
                0AE8          PSZ = PMTSIZ,-
                0AE8          UBF = DEV_NAME,-
                0AE8          USZ = NAME_LEN
                0B2C
                0B2C  INI_FAB:                                    ; Allocate FAB for UETINIDEV
                0B2C          $FAB-
                0B2C          FAC = <GET,PUT,UPD>,-
                0B2C          RAT = CR,-
                0B2C          SHR = <GET,PUT,UPI>,-
                0B2C          FNM = <UETINIDEV.DAT>
                0B7C
                0B7C  INI_RAB:                                    ; Allocate RAB for UETINIDEV
                0B7C          $RAB-
                0B7C          FAB = INI_FAB,-
                0B7C          RBF = BUFFER,-
                0B7C          UBF = BUFFER,-
                0B7C          USZ = REC_SIZE
                0BC0
                0BC0  DDB_RFA:                                    ; RFA storage for INI_RAB
      00000BC6  0BC0          .BLKB   6
                0BC6
                0BC6          .ALIGN  LONG
                0BC8  SUP_FAB:                                    ; Allocate FAB for UETSUPDEV
                0BC8          $FAB-
                0BC8          FAC = GET,-
                0BC8          SHR = <UPI,GET>,-
                0BC8          RAT = CR,-
                0BC8          FOP = UFO,-
                0BC8          FNM = <UETSUPDEV.DAT>
                0C18
                0C18  ;
                0C18  ; Dummy FAB and RAB to copy to the UETP unit blocks
                0C18  ; The following FAB and RAB must be contiguous and in this order!
                0C18  ;
                0C18
                0C18  DUMMY_FAB:
                0C18          $FAB
                0C68
                0C68  DUMMY_RAB:
                0C68          $RAB    RSZ = WRITE_SIZE,-
                0C68                  USZ = READ_SIZE
```

```
                                            B  2
UETDMPF00                  VAX/VMS UETP DEVICE TEST FOR DMP 11/ DMF 16-SEP-1984 01:24:05  VAX/VMS Macro V04-00       Page  15
V04-001                    Main Program                             10-SEP-1984 12:03:55  [UETP.SRC]UETDMPF00.MAR;2       (6)

                                0CAC                 .SBTTL  Main Program
                            00000000                 .PSECT  DMPF,EXE,NOWRT,PAGE
                                0000
                                0000                 .DEFAULT DISPLACEMENT,WORD
                                0000
                       0000     0000     .ENTRY UETDMPF00,^M<>                      ; Entry mask
                                0002
          6D   09D1'CF   DE     0002             MOVAL    SSERROR,(FP)              ; Declare exception handler
                                0007             $SETSFM_S ENBFLG = #1              ; Enable system service failure mode
                                0010             $DCLEXH_S DESBLK = EXIT_DESC       ; Declare an exit handler
                                001B
                                001B             $OPEN    FAB = SYSIN_FAB,-         ; Open SYS$INPUT
                                001B                      ERR = RMS_ERROR
                                002A             $CONNECT RAB = SYSIN_RAB,-         ; Connect RAB to SYS$INPUT
                                002A                      ERR = RMS_ERROR
                   02    E1     0039             BBC      S^#DEV$V_TRM,-            ; BR if SYS$INPUT is NOT a terminal
           1E 0AD8'CF           003B                      SYSIN_FAB+FAB$L_DEV,10$
                                003F             $TRNLOG_S LOGNAM = CONTROLLER,-    ; Allow terminal user to specify...
                                003F                       RSLLEN = DEVNAM_LEN,-    ; ...a logial name...
                                003F                       RSLBUF = DEVDSC          ; ...for the controller to test
              01   50    D1     0058             CMPL     R0,#SS$_NORMAL            ; Was a controller specified?
                   2E    13     005B             BEQL     PROC_CONT_NAME            ; BR if it was - go process it
                                005D     10$:
                                005D             $GET     RAB = SYSIN_RAB,-         ; Read SYS$INPUT...
                                005D                      ERR = RMS_ERROR           ; ...for the controller name
              0B0A'CF    B0     006C             MOVW     SYSIN_RAB+RAB$W_RSZ,-     ; Save the name length
              02E4'CF           0070                      DEVNAM_LEN
                   16    12     0073             BNEQ     PROC_CONT_NAME            ; BR if we got something
         02C6'CF   14    D0     0075             MOVL     #SS$_BADPARAM,STATUS      ; Save an exit status if not
              00C4'CF    DF     007A             PUSHAL   NO_CTRLNAME               ; Prepare for message...
                   01    DD     007E             PUSHL    #1                        ; ...arg count
          00741132 8F    DD     0080             PUSHL    #UETP$_TEXT!STS$K_ERROR   ; ...signal name
                   03    DD     0086             PUSHL    #3                        ; ...arg count
                 0AF6    31     0088             BRW      ERROR_EXIT                ; ...go tell of bad setup
                                008B
                                008B     PROC_CONT_NAME:
     0218'CF   02E4'CF   3C     008B             MOVZWL   DEVNAM_LEN,DEVDSC         ; Set the device name length
               0218'CF   DF     0092             PUSHAL   DEVDSC                    ; Make sure...
               0218'CF   DF     0096             PUSHAL   DEVDSC                    ; ...that the specified controller...
    00000000'GF    02    FB     009A             CALLS    #2,G^STR$UPCASE           ; ...is all uppercase for later comaparison
  52   0218'CF     01    C1     00A1             ADDL3    #1,DEVDSC,R2              ; Estimate the eventual...
       0220'CF     52    A0     00A7             ADDW2    R2,PROCESS_NAME           ; ...process name length (incl. "_")
                         DE     00AC             MOVAL    PROCESS_NAME+8-           ; Locate first available byte...
                                00AD                      +MAX_PROC_NAME-           ; ...in process name handle...
          50   022C'CF          00AD                      -PROCESS_NAME_FREE,R0     ; ...for device name
                   0B    C3     00B1             SUBL3    #PROCESS_NAME_FREE,-      ; Will the device name fit...
              51   52           00B3                      R2,R1                     ; ...in the remaining space?
                   08    15     00B5             BLEQ     10$                       ; BR if it will
              50   51    C2     00B7             SUBL2    R1,R0                       ; Overwrite handle otherwise...
         0220'CF   0F    B0     00BA             MOVW     #MAX_PROC_NAME,PROCESS_NAME ; ...and define the maximum length
                                00BF     10$:
           80   5F 8F    90     00BF             MOVB     #^A/_/,(R0)+              ; Separate handle from device name
  60   0237'CF  0218'CF  28     00C3             MOVC3    DEVDSC,DEV_NAME,(R0)      ; Concatenate handle with device name
                   7E    D4     00CB             CLRL     -(SP)                     ; Set the time stamp flag
              000F'CF    DF     00CD             PUSHAL   TEST_NAME                 ; Set the test name
                   02    DD     00D1             PUSHL    #2                        ; Push the argument count
          00741039 8F    DD     00D3             PUSHL    #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
```

C 2

```
00000000'GF   04   FB  00D9          CALLS   #4,G^LIB$SIGNAL               ; Print the startup message
     0002'CF   08   A8  00E0          BISW2   #BEGIN_MSGM,FLAG              ; Set flag so we don't print it again
                        00E5          $SETPRN_S PRCNAM = PROCESS_NAME ; Set the process name to UETDMPF00_x
                        00F0
           02   E1  00F0          BBC     S^#DEV$V_TRM,-                ; BR if SYS$INPUT is NOT a terminal
   66 0AD8'CF            00F2                  SYSIN_FAB+FAB$L_DEV,20$
                        00F6          $GETDVI_S DEVNAM = SYS$INPUT,-    ; Get the name of...
                        00F6                  EFN    = #SS_SYNCH_EFN,- ; ...device which may abort test
                        00F6                  ITMLST = INPUT_ITMLST,-
                        00F6                  IOSB   = QUAD_STATUS
   45 02CA'CF   E9  0112          BLBC    QUAD_STATUS,20$               ; Avoid CTRL/C handler if any error
                        0117          $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C AST handler
                        0117                  CHAN   = TTCHAN
                        0128          $QIOW_S CHAN    = TTCHAN,-        ; Enable CTRL/C AST's...
                        0128                  FUNC    = #IO$_SETMODE!IO$M_CTRLCAST,-
                        0128                  P1      = CCASTHAND
       0220'CF   DF  0149          PUSHAL  PROCESS_NAME                  ; ...and tell the user...
           01   DD  014D          PUSHL   #1                            ; ...
   0074832B 8F   DD  014F          PUSHL   #UETP$_ABORTC!STS$K_SUCCESS ; ...how to abort gracefully...
00000000'GF   03   FB  0155          CALLS   #3,G^LIB$SIGNAL               ; ...
                        015C     20$:
```

```
                                    015C  ;
                                    015C  ; From UETINIDEV.DAT and UETSUPDEV.DAT, get information which gives controller
                                    015C  ; and unit configuration and lets us know if the setup to run this test was
                                    015C  ; done correctly.
                                    015C  ;
                                    015C          $OPEN   FAB = INI_FAB,-           ; Open file 'UETINIDEV.DAT''
                                    015C                  ERR = RMS_ERROR
                                    016B          $CONNECT RAB = INI_RAB,-          ; Connect the RAB and FAB
                                    016B                  ERR = RMS_ERROR
                                    017A          $MGBLSC_S  INADR = INADDRESS,-    ; Connect to UETSUPDEV global section
                                    017A                  RETADR = OUTADDRESS,-
                                    017A                  GSDNAM = SUPDEV_GBLSEC,-
                                    017A                   FLAGS = #SEC$M_EXPREG
          00000978 8F     50    D1  0199          CMPL    R0,#SS$_NOSUCHSEC            ; Was the section already there?
                          37    12  01A0          BNEQ    30$                          ; BR if it was...
                                    01A2          $OPEN   FAB = SUP_FAB,-              ; ...else open 'UETSUPDEV.DAT''
                                    01A2                  ERR = RMS_ERROR
                                    01B1          $CRMPSC_S CHAN = SUP_FAB+FAB$L_STV,- ; Create the global section
                                    01B1                  INADR = INADDRESS,-
                                    01B1                  RETADR = OUTADDRESS,-
                                    01B1                  GSDNAM = SUPDEV_GBLSEC,-
                                    01B1                   FLAGS = #SEC$M_EXPREG!SEC$M_GBL
                                    01D9  30$:
     56   02DE'CF    02DA'CF    C3  01D9          SUBL3   OUTADDRESS,OUTADDRESS+4,R6 ; Compute global section length
                                    01E1
                                    01E1  FIND_IT:
                                    01E1          $GET    RAB = INI_RAB,-           ; Get the first record
                                    01E1                  ERR = RMS_ERROR
                 01F5'CF        DF  01F0          PUSHAL  CONT_DESC                 ; Make sure...
                 01F5'CF        DF  01F4          PUSHAL  CONT_DESC                 ; ...that the controller name...
          00000000'GF     02    FB  01F8          CALLS   #2,G^STR$UPCASE           ; ...is all uppercase letters
             0014'CF   44 8F    91  01FF          CMPB    #^A/D/,BUFFER             ; Is this a DDB?
                          27    13  0205          BEQL    10$                       ; Go on if not
             0014'CF   45 8F    91  0207          CMPB    #^A/E/,BUFFER             ; Is this the end of the file?
                          D2    12  020D          BNEQ    FIND_IT                   ; Continue on if not
                 0218'CF        DF  020F          PUSHAL  DEVDSC                    ; Push device not supported message
                 0220'CF        DF  0213          PUSHAL  PROCESS_NAME              ; Parameters on the stack
                          02    DD  0217          PUSHL   #2
            00748333 8F         DD  0219          PUSHL   #UETP$_DENOSU
                          02    F0  021F          INSV    #STS$K_ERROR,-            ; Set the severity code...
                          00        0221                  #STS$V_SEVERITY,-
                       6E 03        0222                  #STS$S_SEVERITY,(SP)
             02C6'CF      6E    D0  0224          MOVL    (SP),STATUS               ; ...and save it as the exit status
                          04    DD  0229          PUSHL   #4
                        0953    31  022B          BRW     ERROR_EXIT                ; Exit in error
                                    022E  10$:
 0237'CF   001A'CF    02E4'CF   29  022E          CMPC    DEVNAM_LEN,BUFFER+6,DEV_NAME    ; Is this the right controller?
                          A7    12  0238          BNEQ    FIND_IT                   ; BR if not
     0BC0'CF    0B8C'CF   06    28  023A          MOVC3   #6,INI_RAB+RAB$W_RFA,DDB_RFA ; Save the Record File Address
             0018'CF   54 8F    91  0242          CMPB    #^A/T/,BUFFER+4           ; Can we test this controller?
                          2F    13  0248          BEQL    FOUND_IT                  ; BR if we can...
                                    024A          $FAO_S  CTRSTR = DEAD_CTRLNAME,- ; ...and yell at user if we can't
                                    024A                  OUTLEN = BUFFER_PTR,-
                                    024A                  OUTBUF = FAO_BUF,-
                                    024A                  P1     = #DEVDSC
             02C6'CF      14    D0  0263          MOVL    #SS$_BADPARAM,STATUS      ; Set return status
                 000C'CF        DF  0268          PUSHAL  BUFFER_PTR                ; ...
```

```
                                                        E 2
UETDMPF00                     VAX/VMS UETP DEVICE TEST FOR DMP 11/ DMF 16-SEP-1984 01:24:05  VAX/VMS Macro V04-00          Page  18
V04-001                       Main Program                            10-SEP-1984 12:03:55  [UETP.SRC]UETDMPF00.MAR;2           (7)

                            01  DD  026C           PUSHL   #1                          ; ...
                  00741132  8F  DD  026E           PUSHL   #UETP$_TEXT!STS$K_ERROR     ; ...
                            03  DD  0274           PUSHL   #3                          ; ...
                          0908  31  0276           BRW     ERROR_EXIT                  ; We can't test what we can't test
                                    0279
                                    0279   FOUND_IT:
                                    0279           $GET    RAB = INI_RAB,-             ; Get a record
                                    0279                   ERR = RMS_ERROR
                       01F5'CF  DF  0288           PUSHAL  CONT_DESC                   ; Make sure...
                       01F5'CF  DF  028C           PUSHAL  CONT_DESC                   ; ...that this line...
              00000000'GF   02  FB  0290           CALLS   #2,G^STR$UPCASE             ; ...is all uppercase letters
                 0014'CF 55 8F  91  0297           CMPB    #^A/U/,BUFFER               ; Is this a UCB?
                            24  13  029D           BEQL    30$                         ; BR if it is
                 0014'CF 44 8F  91  029F           CMPB    #^A/D/,BUFFER               ; Is this a DDB?
                            19  13  02A5           BEQL    20$                         ; BR if yes
                 0014'CF 45 8F  91  02A7           CMPB    #^A/E/,BUFFER               ; Is this the end?
                            11  13  02AD           BEQL    20$                         ; BR if yes
                                    02AF   10$:
                       0151'CF  DF  02AF           PUSHAL  ILLEGAL_REC                 ; Then this is an error in the record
                            01  DD  02B3           PUSHL   #1                          ; Push the error message
                  00741132  8F  DD  02B5           PUSHL   #UETP$_TEXT!STS$K_ERROR     ; Push the signal name
                            03  DD  02BB           PUSHL   #3                          ; Push the temp arg count
                          08C1  31  02BD           BRW     ERROR_EXIT                  ; Finish for good
                                    02C0   20$:
                          0123  31  02C0           BRW     ALL_SET                     ; Found DDB or END
                                    02C3   30$:
                 0018'CF 54 8F  91  02C3           CMPB    #^A/T/,BUFFER+4             ; Is the unit testable?
                            AE  12  02C9           BNEQ    FOUND_IT                    ; BR if not
                            01  DD  02CB           PUSHL   #1                          ; Flag to ignore blanks when converting
                            02  DD  02CD           PUSHL   #2                          ; Set byte size of results
                       02E2'CF  DF  02CF           PUSHAL  UNIT_NUMBER                 ; Set address to receive word
                       01ED'CF  DF  02D3           PUSHAL  UNIT_DESC                   ; Push string address
              00000000'GF   04  FB  02D7           CALLS   #4,G^OTS$CVT_TI_L           ; Convert ASCII unit # to decimal
                         CE 50  E9  02DE           BLBC    R0,10$                      ; Don't allow bogus unit to pass
                         05 20  3B  02E1           SKPC    #^A/ /,#MAX_UNIT_DESIG,-    ; Find out where unit number really is
                       001A'CF      02E4                   BUFFER+6
                            50  D7  02E7           DECL    R0                          ; Units must all be at least one digit
                   61  50   30  3B  02E9           SKPC    #^A/0/,R0,(R1)              ; Skip leading zeroes on the unit
                            50  D6  02ED           INCL    R0                          ; Compensate for DECL above
         0218'CF   02E4'CF  50  A1  02EF           ADDW3   R0,DEVNAM_LEN,DEVDSC        ; Calculate device'unit string length
                   52  02E4'CF  3C  02F7           MOVZWL  DEVNAM_LEN,R2               ; Offset to unit number in DEVDSC
            0237'C2  61     50  28  02FC           MOVC3   R0,(R1),DEV_NAME(R2)        ; Append unit number to device
                                    0302           $GETDEV_S DEVNAM = DEVDSC,-         ; Get the device characteristics
                                    0302                     PRIBUF = DIB
                   57  0252'CF  9A  0317           MOVZBL  DIBBUF+DIB$B_DEVCLASS,R7    ; Save the device class
                   58  0253'CF  9A  031C           MOVZBL  DIBBUF+DIB$B_DEVTYPE,R8     ; Save the device type
                                    0321           $FAO_S  CTRSTR = CS1,-
                                    0321                   OUTBUF = FAO_BUF,-
                                    0321                   P1     = R7,-
                                    0321                   P2     = R8                 ; Make it into a string
  02DA'DF   56   0014'CF    06  39  0336           MATCHC  #6,BUFFER,R6,@OUTADDRESS    ; Find the device class and type
                            1E  13  033F           BEQL    40$                         ; BR if it was found
                                    0341           $FAO_S  CTRSTR = CS3,-              ; Try for full class support
                                    0341                   OUTBUF = FAO_BUF,-
                                    0341                       P1 = R7
  02DA'DF   56   0014'CF    06  39  0354           MATCHC  #6,BUFFER,R6,@OUTADDRESS    ; Find the device class only
                            0D  12  035D           BNEQ    50$                         ; BR if not found
                          035F     40$:
      55   000F'CF    9A  035F               MOVZBL  TEST_NAME,R5             ; Get the test name length
0017'CF    63     55  29  0364               CMPC3   R5,(R3),TEST_NAME+8      ; Are we the right test?
                  1F  13  036A               BEQL    60$                      ; BR if yes
                          036C     50$:
             0218'CF  DF  036C               PUSHAL  DEVDSC                   ; Push device not supported message
             0220'CF  DF  0370               PUSHAL  PROCESS_NAME             ; Parameters on the stack
                  02  DD  0374               PUSHL   #2                       ; Push the argument count
         00748333 8F  DD  0376               PUSHL   #UETP$_DENOSU
                  02  F0  037C               INSV    #STS$K_ERROR,-
                  00      037E                       #STS$V_SEVERITY,-
              6E  03      037F                       #STS$S_SEVERITY,(SP)     ; Set the severity code...
         02C6'CF  6E  D0  0381               MOVL    (SP),STATUS              ; ...and save it as the exit status
                  04  DD  0386               PUSHL   #4                       ; Push the partial arg count...
                07F6  31  0388               BRW     ERROR_EXIT               ; ...and split this scene
```

```
0388  ;+
0388  ;       The following code dynamically allocates enough memory for a unit block,
0388  ;       a device dependent parameter area and I/O buffers. The unit block is inserted
0388  ;       into the queue header UNIT_LIST.  It then initializes the unit block.
0388  ;       A comment indicates where the device dependent parameters should be
0388  ;       initialized.  The unit block format is as follows:
0388  ;
0388  ;                               +---------------+ -------
0388  ;               UETUNT$L_FLINK  !               !       ^
0388  ;                               +---------------+       !
0388  ;               UETUNT$L_BLINK  !               !       !
0388  ;                               +-----------+---+       !
0388  ;               UETUNT$B_TYPE               !   !       !
0388  ;                                   +-------+---+       !
0388  ;               UETUNT$W_SIZE       !   !   !   contains DEVDEP_SIZE + UETUNT$C_INDSIZ
0388  ;                               +---+-------+           !
0388  ;               UETUNT$B_FLAGS  !   !                   !
0388  ;                               +---+   +-------+       !
0388  ;               UETUNT$W_CHAN           !   !   !       !
0388  ;                               +-------+-------+       !
0388  ;               UETUNT$W_FUNC   !   !   !               !
0388  ;                               +-------+-------+       +----- UETUNT$C_SIZE
0388  ;               UETUNT$L_ITER   !               !       !
0388  ;                               +---------------+       !
0388  ;               UETUNT$T_FILSPC !               !       !
0388  ;                               !/\/\/\/\/\/\/\/!       !
0388  ;                               NAM$C_MAXRSS bytes      !
0388  ;                               !/\/\/\/\/\/\/\/!       !
0388  ;                               !               !       !
0388  ;               UETUNT$K_FAB    !---------------!       !
0388  ;                               !               !       !
0388  ;                               !/\/\/\/\/\/\/\/!       !
0388  ;                                FAB$C_BLN bytes        !
0388  ;                               !/\/\/\/\/\/\/\/!       !
0388  ;                               !               !       !
0388  ;               UETUNT$K_RAB    +---------------+       !
0388  ;                               !               !       !
0388  ;                               !\/\/\/\/\/\/\/\!       !
0388  ;                                RAB$C_BLN bytes        !
0388  ;                               !\/\/\/\/\/\/\/\!       !
0388  ;                               !               !       v
0388  ;               UETUNT$K_DEVDEP +---------------+ -------
0388  ;                               !               !       ^
0388  ;                               !\/\/\/\/\/\/\/\!       !
0388  ;                               !  user defined !       +----- DEVDEP_SIZE
0388  ;                               !\/\/\/\/\/\/\/\!       !
0388  ;                               !               !       v
0388  ;                               +---------------+ -------
0388  ;            READ/WRITE buffers !               !       ^
0388  ;                               !\/\/\/\/\/\/\/\!       !
0388  ;                               !  user defined !       +----- WRITE_SIZE and READ_SIZE
0388  ;                               !/\/\/\/\/\/\/\/!       !
0388  ;                               !               !       !
0388  ;                               !               !       v
0388  ;                               +---------------+ -------
0388  ;-
```

```
                            038B      60$:
                            038B            $EXPREG_S PAGCNT = #PAGES,-        ; Get a new node of demand zero memory
                            038B                      RETADR = NEW_NODE
 0A88'CF   0A90'DF    5D    039C            INSQTI  @NEW_NODE,UNIT_LIST        ; Put the new node in the unit list
      56   0A90'CF    D0    03A3            MOVL    NEW_NODE,R6                ; Save a copy of its address
       08 A6    01    90    03A8            MOVB    #1,UETUNT$B_TYPE(R6)       ; Set the structure type
           01A4 8F    B0    03AC            MOVW    #UETUNT$C_INDSIZ+DEVDEP_SIZE,-
             09 A6          03B0                    UETUNT$W_SIZE(R6)          ; Set the structure size
   14 A6   0218'CF    90    03B2            MOVB    DEVDSC,UETUNT$T_FILSPC(R6) ; Set the device name size
 021C'DF   0218'CF    28    03B8            MOVC3   DEVDSC,@DEVDSC+4,-
             15 A6          03BF                    UETUNT$T_FILSPC+1(R6)      ; Save the device name
           0094 8F    28    03C1            MOVC3   #FAB$C_BLN+RAB$C_BLN,-
 0110 C6   0C18'CF          03C5                    DUMMY_FAB,UETUNT$C_FAB(R6) ; Save a FAB and a RAB away
      57   0110 C6    DE    03CB            MOVAL   UETUNT$K_FAB(R6),R7        ; Save the FAB address
      58   0160 C6    DE    03D0            MOVAL   UETUNT$K_RAB(R6),R8        ; Save the RAB address
     3C A8      57    D0    03D5            MOVL    R7,RAB$L_FAB(R8)           ; Set the FAB address in the RAB
             14 A6    90    03D9            MOVB    UETUNT$T_FILSPC(R6),-
             34 A7          03DC                    FAB$B_FNS(R7)              ; Set the FNS field in the FAB
             15 A6    DE    03DE            MOVAL   UETUNT$T_FILSPC+1(R6),-
             2C A7          03E1                    FAB$L_FNA(R7)              ; Set the FNA field in the FAB
                            03E3      ;
                            03E3      ; Set the device dependent parameters in here
                            03E3      ;
            FE93      31    03E3            BRW     FOUND_IT                   ; Do the next UCB
```

```
                         03E6      ;
                         03E6      ; Arrive here when we have the device configuration.  In normal or loop forever
                         03E6      ; mode, set a timer far enough in the future such that we can do a reasonable
                         03E6      ; set of tests before the timer expires, but if our device gets hung, the
                         03E6      ; program won't waste too much time before noticing.  Let one-shot mode be a
                         03E6      ; special case.
                         03E6      ;
                         03E6      ALL_SET:
         0A88'CF    D5   03E6              TSTL    UNIT_LIST                 ; Anything to test?
              16    12   03EA              BNEQ    10$                       ; BR if yes
         012B'CF    DF   03EC              PUSHAL  NOUNIT_SELECTED           ; Else set up the error message...
              01    DD   03F0              PUSHL   #1                        ; ...argument count...
     00741132 8F    DD   03F2              PUSHL   #UETP$_TEXT!STS$K_ERROR   ; ...signal name...
              03    DD   03F8              PUSHL   #3                        ; ...and parameter count
   02C6'CF    14    D0   03FA              MOVL    #SS$_BADPARAM,STATUS      ; Set return status
            077F    31   03FF              BRW     ERROR_EXIT                ; ...and give up, complaining
                         0402      10$:
   0002'CF    04    A8   0402              BISW2   #SAFE_TO_UPDM,FLAG        ; OK safe to update UETINIDEV.DAT now
```

```
                              0407          .SBTTL  Test the DMP/DMF
                              0407
                              0407  START_TEST:
                              0407
                              0407          $ASSIGN_S -                          ; Assign channel to the device
                              0407                  DEVNAM = DEVDSC,-
                              0407                  CHAN = XD_CHAN
                              0418
                              0418
             22 50      E8    0418          BLBS    R0,10$                       ; BR if no failure
     02C6'CF    50      D0    041B          MOVL    R0,STATUS                    ; Save the failure status
        02C6'CF         DD    0420          PUSHL   STATUS                       ; Push the error code...
        02C6'CF         DD    0424          PUSHL   STATUS
        0218'CF         DF    0428          PUSHAL  DEVDSC                       ; ...and the device designation...
        000F'CF         DF    042C          PUSHAL  TEST_NAME                    ; ...and the test name...
             03         DD    0430          PUSHL   #3                           ; ...and the arg count...
    0074819A 8F         DD    0432          PUSHL   #UETP$_DEUNUS!STS$K_ERROR ; ...and the signal name...
             06         DD    0438          PUSHL   #6                           ; ...and the total argument count...
           0744         31    043A          BRW     ERROR_EXIT                   ; ...and bail out completely
                              043D  10$:
                              043D  ;
                              043D  ; Set up P1 device char buffer, P2 buffer is set up in Read/write section
                              043D  ;
                              043D  RESTART:
 53   0388'CF           DE    043D          MOVAL   P1BUF+2,R3                           ; Address of device char for p1
 83   0200 8F           B0    0442          MOVW    #MAX_MSG_LEN,(R3)+                   ; Maximum message length
      63   02           90    0447          MOVB    #XM$M_CHR_LOOPB,(R3)                 ; Set loop back mode in char
                              044A
                              044A          $SETIMR_S -                                  ; Set up half minute timer
                              044A                  DAYTIM = HALFMIN,-                   ; to prevent hung
                              044A                  ASTADR = TIME_ERR_OUT,-
                              044A                  REQIDT = #START_TO_MSG
                              0461  START_CONT:
                              0461          $QIOW_S -                            ; Start the controller
                              0461                  CHAN = XD_CHAN,-
                              0461                  FUNC = #IO$_SETMODE!IO$M_CTRL!IO$M_STARTUP,-
                              0461                  IOSB = XD_IOSB,-
                              0461                  ASTADR = CHK_QIO_AST,-
                              0461                  ASTPRM = #START_CONT_PRM,-
                              0461                  P1 = P1BUF,-
                              0461                  P2 = #P2BUF_DESC,-
                              0461                  P3 = #RECVPOOL_SIZ
                              0492
                              0492  START_TRI:
                              0492          $QIOW_S -                            ; Start the tributary
                              0492                  CHAN = XD_CHAN,-
                              0492                  FUNC = #IO$_SETMODE!IO$M_STARTUP,-
                              0492                  IOSB = XD_IOSB,-
                              0492                  ASTADR = CHK_QIO_AST,-
                              0492                  ASTPRM = #START_TRIB_PRM,-
                              0492                  P1 = TR_P1BUF,-
                              0492                  P2 = #TR_P2BUF_DESC,-
                              0492                  P3 = #RECVPOOL_SIZ   ; Common receive pool = 4 buffer
                              04C3
0002'CF    20           A8    04C3          BISW2   #FLAG_SHUTDNM,FLAG           ; Set flag to say shut down the
                              04C8                                               ; device if errors occur
                              04C8
```

```
                             04C8          $CANTIM_S REQIDT = #START_TO_MSG ; Cancel hung timer
                             04D7
                             04D7          $TRNLOG_S LOGNAM = MODE,-        ; Get the run mode
                             04D7                    RSLLEN = BUFFER_PTR,-
                             04D7                    RSLBUF = FAO_BUF
                             04F0
        0014'CF    20    8A  04F0          BICB2   #LC_BITM,BUFFER          ; Convert to upper case
      0014'CF   4F 8F    91  04F5          CMPB    #^A7O/,BUFFER            ; Is this a one shot?
                   0D    12  04FB          BNEQ    10$                      ; BR if not
        0002'CF    02    A8  04FD          BISW2   #TEST_OVERM,FLAG         ; End after one iteration
        0002'CF    10    A8  0502          BISW2   #MODE_IS_ONEM,FLAG       ; Set mode is "ONE" flag
              0013       31  0507          BRW     LOOPBACK_TEST            ; Skip the 3 min timer, mode is "one"
                             050A  10$:                                     ; Not one shot
                             050A          $SETIMR_S DAYTIM = THREEMIN,-    ; Set 3 minutes timer for xmit/recv
                             050A                    ASTADR = TIME_SUC_OUT  ; The test will do xmit/recv for about
                             051D                                           ; 3 minutes
                             051D  ;
                             051D  ; Loopback test transmit and receive random data with different message length
                             051D  ;
                             051D  LOOPBACK_TEST:
           52   AA 8F    9A  051D          MOVZBL  #^XAA,R2                 ; Random number 1
              53    2E   9A  0521          MOVZBL  #^X2E,R3                 ; Random number 2
     57  00000200 8F     D0  0524          MOVL    #MAX_MSG_LEN,R7          ; Maximum message length
                             052B  SET_XMIT_BUF:                            ; Set up transmit buffer
         56   0680'CF    DE  052B          MOVAL   XMIT_BUF,R6              ; Transmit buffer address
              54    57   D0  0530          MOVL    R7,R4                    ; Message length in bytes
                             0533  10$:
              52    53   C0  0533          ADDL2   R3,R2                    ; Random number as data
              86    52   90  0536          MOVB    R2,(R6)+                 ; Fill in the transmit buffer
              F7    54   F5  0539          SOBGTR  R4,10$                   ; Branch if more bytes to be filled
                             053C
                             053C          $SETIMR_S -                      ; Set half minute timer to prevent hung
                             053C                    DAYTIM = HALFMIN,-
                             053C                    ASTADR = TIME_ERR_OUT,-
                             053C                    REQIDT = #RW_TO_MSG
                             0553
              58    10   D0  0553          MOVL    #LIMIT,R8                ; Loop 16 times for each msg length
                             0556  XMIT:
                             0556          $QIO_S -                         ; Transmit data message
                             0556                    EFN = #XMIT_EFN,-      ; Event flag
                             0556                    CHAN = XD_CHAN,-       ; Channel
                             0556                    FUNC = #IO$_WRITEVBLK,- ; Transmit
                             0556                    IOSB = XD_IOSB,-       ; IOSB
                             0556                    ASTADR = CHK_QIO_AST,- ; Completion ast routine
                             0556                    ASTPRM = #WRITE_PRM,-  ; Ast parameter
                             0556                    P1 = XMIT_BUF,-        ; Addr of transmit buffer
                             0556                    P2 = R7                ; message length in bytes
                             0581
                             0581  RECV:
                             0581          $QIOW_S -                        ; Read data message
                             0581                    EFN = #RECV_EFN,-      ; Event flag
                             0581                    CHAN = XD_CHAN,-       ; Channel
                             0581                    FUNC = #IO$_READVBLK,- ; Receive message
                             0581                    IOSB = RCV_IOSB,-      ; IOSB
                             0581                    ASTADR = RECV_AST,-    ; Completion ast to check data received
                             0581                    ASTPRM = R7,-          ; Ast parameter = message length
                             0581                    P1 = RECV_BUF,-        ; Receive buffer
```

```
                          05B1                 P2 = R7                  ; Message length in bytes
                          05A8
           02E6'CF   D6   05A8         INCL    ITERATION                ; Increment iteration count
                          05AC
                          05AC         $WAITFR_S EFN = #XMIT_EFN        ; Wait until transmit done
                          05B5
             9E 58   F5   05B5         SOBGTR  R8,XMIT                  ; Loop for 16 times
                          05B8
                          05B8         $CANTIM_S -                      ; Cancel hung timer
                          05B8                 REQIDT = #RW_TO_MSG
                          05C7
   09 0002'CF   01   E0   05C7         BBS     #TEST_OVERV,FLAG,SENSE_TEST ; Is the test over?
             03 57   F5   05CD         SOBGTR  R7,10$                   ; Decrement message length by one and
                          05D0                                          ; try again
              FF4A   31   05D0         BRW     LOOPBACK_TEST            ; Re-try from beginning
              FF55   31   05D3  10$:   BRW     SET_XMIT_BUF             ; Set  new data in tranmit buffer
                          05D6
                          05D6  SENSE_TEST:
                          05D6         $QIOW_S -                        ; Read device (trib.) charracteristic
                          05D6                 CHAN = XD_CHAN,-
                          05D6                 FUNC = #IO$_SENSEMODE,-
                          05D6                 IOSB = XD_IOSB,-
                          05D6                 P1 = SENSE_P1BUF,-
                          05D6                 P2 = #SENSE_P2DESC
                          05FD
                          05FD         $FAO_S  CTRSTR = SENSE_PRM,-
                          05FD                 OUTLEN = ALT_BUFFER_PTR,-
                          05FD                 OUTBUF = ALT_FAO_BUF,-
                          05FD                 P1     = #DEVDSC
           0116'CF   DF   0616         PUSHAL  ALT_BUFFER_PTR
      08A5'CF   01   FB   061A         CALLS   #1,CHECK_IOSB            ; Check status
                          061F
      54   0672'CF   3C   061F         MOVZWL  XD_IOSB+2,R4             ; Number of bytes returned for p2 buff
      55   03B2'CF   DE   0624         MOVAL   TR_P2BUF,R5              ; Address of P2 buff
           57   06   D0   0629         MOVL    #TR_P2BUF_LEN,R7         ; P2 length
                          062C  10$:
      56   03C8'CF   DE   062C         MOVAL   SENSE_P2BUF,R6           ; Address of P2 buff returned
 66   54   65   06   39   0631         MATCHC  #6,(R5),R4,(R6)          ; Check the parameters returned
                0C   12   0636         BNEQ    30$                      ; Br if not match
        55   06 A5   DE   0638         MOVAL   6(R5),R5                 ; Next parameter
           57   06   C2   063C         SUBL2   #6,R7                    ; Index
                EB   12   063F         BNEQ    10$                      ; Br if more parameters to check
              001F   31   0641         BRW     ERROR_TEST               ; Otherwise go to test error case
                          0644  30$:
                          0644         $FAO_S  CTRSTR = SENSE_ERRMSG,-
                          0644                 OUTLEN = BUFFER_PTR,-
                          0644                 OUTBUF = FAO_BUF,-
                          0644                 P1     = (R5),-
                          0644                 P2     = 2(R5)
           000C'CF   DF   065C         PUSHAL  BUFFER_PTR               ; Error message
              0235   31   0660         BRW     FAIL_OUT                 ; Failure exit
                          0663
                          0663  ERROR_TEST:
                          0663         $SETSFM_S ENBFLG = #0            ; Turn off system service mode
                          066C
                          066C  ;
                          066C  ; Read data with IO$M_NOW specified but no data available
```

```
                                066C      ;
                                066C              $QIOW_S -                           ; Read data message
                                066C                      CHAN = XD_CHAN,-
                                066C                      FUNC = #IO$_READVBLK!IO$M_NOW,-
                                066C                      IOSB = XD_IOSB,-
                                066C                      P1 = RECV_BUF,-
                                066C                      P2 = #128
                                0693
      57   00000870 8F    D0    0693              MOVL    #SS$_ENDOFFILE,R7
         58   0670'CF     3C    069A              MOVZWL  XD_IOSB,R8
              57     58   B1    069F              CMPW    R8,R7                       ; Correct error code?
                     3B   12    06A2              BNEQ    ERRTST_ERR                  ; Br if not
                                06A4
                                06A4      ;
                                06A4      ; Buffer not enough to hold all information from IO$_SENSEMODE
                                06A4      ;
                                06A4
                                06A4              $QIOW_S -                           ; Read device (trib. ) charracteristic
                                06A4                      CHAN = XD_CHAN,-
                                06A4                      FUNC = #IO$_SENSEMODE,-
                                06A4                      IOSB = XD_IOSB,-
                                06A4                      P1 = SENSE_P1BUF,-
                                06A4                      P2 = #ERRTST_P2DESC
                                06CB
      57   00000601 8F    D0    06CB              MOVL    #SS$_BUFFEROVF,R7
         58   0670'CF     3C    06D2              MOVZWL  XD_IOSB,R8
              57     58   B1    06D7              CMPW    R8,R7                       ; Error code = buffer overflow?
                     03   12    06DA              BNEQ    ERRTST_ERR                  ; Error if not
                   0099   31    06DC              BRW     READ_ERRCOUNT               ; Br to read and clear error count
                                06DF
                                06DF      ERRTST_ERR:
              04AF'CF     28    06DF              MOVC3   COMP_STATUS_MSG,-           ; We need an error message...
   0014'CF    04B7'CF           06E3                      COMP_STATUS_MSG+8,BUFFER
              04AF'CF     A3    06E9              SUBW3   COMP_STATUS_MSG,-           ; ...to compare...
        59    00FA 8F           06ED                      #TEXT_BUFFER,R9
       000C'CF    59      3C    06F1              MOVZWL  R9,BUFFER_PTR
       0010'CF    53      D0    06F6              MOVL    R3,BUFFER_PTR+4
                                06FB              $GETMSG_S MSGID = R7,-              ; ...the error we expected...
                                06FB                        MSGLEN = BUFFER_PTR,-
                                06FB                        BUFADR = BUFFER_PTR
   0010'CF    000C'CF     C0    0710              ADDL2   BUFFER_PTR,BUFFER_PTR+4
        59    000C'CF     A2    0717              SUBW2   BUFFER_PTR,R9
              04E7'CF     28    071C              MOVC3   RECEIVED_MSG,-
              04EF'CF           0720                      RECEIVED_MSG+8,-
              0010'DF           0723                      @BUFFER_PTR+4
       0010'CF    53      D0    0726              MOVL    R3,BUFFER_PTR+4
        59    04E7'CF     A2    072B              SUBW2   RECEIVED_MSG,R9
       000C'CF    59      3C    0730              MOVZWL  R9,BUFFER_PTR
                                0735              $GETMSG_S MSGID = R8,-              ; ...with the one we received
                                0735                        MSGLEN = BUFFER_PTR,-
                                0735                        BUFADR = BUFFER_PTR
        59    000C'CF     A2    074A              SUBW2   BUFFER_PTR,R9
   59    00FA 8F    59    A3    074F              SUBW3   R9,#TEXT_BUFFER,R9
              59    59    3C    0755              MOVZWL  R9,R9
   0014'C9    2E22 8F     B0    0758              MOVW    #^A/"./,BUFFER(R9)
   000C'CF    59    02    A1    075F              ADDW3   #2,R9,BUFFER_PTR
   0010'CF    0014'CF     DE    0765              MOVAL   BUFFER,BUFFER_PTR+4
```

```
        000C'CF    DF  076C  1080              PUSHAL  BUFFER_PTR                ; Error message
   02C6'CF    58   D0  0770  1081              MOVL    R8,STATUS                 ; Save our actual error as exit status
           0120    31  0775  1082              BRW     FAIL_OUT                  ; Failure exit
                       0778  1083
                       0778  1084 READ_ERRCOUNT:
                       0778  1085              $SETSFM_S ENBFLG = #1             ; Turn on  system service mode
                       0781  1086
                       0781  1087              $QIOW_S -                         ; Read and clear the error counters
                       0781  1088                      CHAN = XD_CHAN,-
                       0781  1089                      FUNC = #IO$_SENSEMODE!IO$M_RD_COUNT!IO$M_CLR_COUNT,-
                       0781  1090                      IOSB = XD_IOSB,-
                       0781  1091                      P2 = #ERRCOUNT_DESC
                       07A6  1092
                       07A6  1093              $FAO_S  CTRSTR = SENSE_PRM,-
                       07A6  1094                      OUTLEN = ALT_BUFFER_PTR,-
                       07A6  1095                      OUTBUF = ALT_FAO_BUF,-
                       07A6  1096                      P1     = #DEVDSC
        0116'CF    DF  07BF  1097              PUSHAL  ALT_BUFFER_PTR
   08A5'CF    01   FB  07C3  1098              CALLS   #1,CHECK_IOSB             ; Check status
                       07C8  1099
                       07C8  1100 CLEAN_EXIT:
                       07C8  1101
   0002'CF    20   AA  07C8  1102              BICW2   #FLAG_SHUTDNM,FLAG        ; Clear the shutdown flag
                       07CD  1103
                       07CD  1104              $QIOW_S -                         ; Shut down the device
                       07CD  1105                      CHAN = XD_CHAN,-
                       07CD  1106                      FUNC = #IO$_SETMODE!IO$M_CTRL!IO$M_SHUTDOWN,-
                       07CD  1107                      IOSB = XD_IOSB
                       07EE  1108
                       07EE  1109              $FAO_S  CTRSTR = SET_PRM,-
                       07EE  1110                      OUTLEN = ALT_BUFFER_PTR,-
                       07EE  1111                      OUTBUF = ALT_FAO_BUF,-
                       07EE  1112                      P1     = #DEVDSC
        0116'CF    DF  0807  1113              PUSHAL  ALT_BUFFER_PTR
   08A5'CF    01   FB  080B  1114              CALLS   #1,CHECK_IOSB             ; Check status
                       0810  1115
                       0810  1116 SUC_EXIT:
                       0810  1117              $TRNLOG_S LOGNAM = MODE,-
                       0810  1118                        RSLLEN = BUFFER_PTR,-
                       0810  1119                        RSLBUF = FAO_BUF        ; Get the run mode
   0014'CF    20   8A  0829  1120              BICB2   #LC_BITM,BUFFER           ; Convert to upper case
 0014'CF   4C 8F   91  082E  1121              CMPB    #^A/L/,BUFFER             ; Is this a loop for ever?
             40    12  0834  1122              BNEQ    10$                       ; BR if not
   0002'CF    02   AA  0836  1123              BICW2   #TEST_OVERM,FLAG          ; Reset the termination flag
        02EA'CF    D6  083B  1124              INCL    PASS                      ; Bump the pass count
                       083F  1125              $FAO_S  CTRSTR = PASS_MSG,-
                       083F  1126                      OUTLEN = BUFFER_PTR,-
                       083F  1127                      OUTBUF = FAO_BUF,-
                       083F  1128                      P1     = PASS,-
                       083F  1129                      P2     = ITERATION,-
                       083F  1130                      P3     = #0              ; Make the end of pass message
        000C'CF    DF  085C  1131              PUSHAL  BUFFER_PTR                ; Push the string desc.
             01    DD  0860  1132              PUSHL   #1                        ; Push arg count
   00741133 8F     DD  0862  1133              PUSHL   #UETP$_TEXT!STS$K_INFO    ; Push the signal name
 00000000'GF  03   FB  0868  1134              CALLS   #3,G^LIB$SIGNAL           ; Print the end of pass message
        02E6'CF    D4  086F  1135              CLRL    ITERATION                 ; Reset the iteration count
           FBC7    31  0873  1136              BRW     RESTART                   ; Do the next pass
```

```
                                 0876       10$:
   56  0A88'CF   00000A88'8F  C1 0876                ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; Set the unit block list header
                        02    88 0880                BISB2   #UETUNT$M_TESTABLE,-
                     0B A6       0882                        UETUNT$B_FLAGS(R6)       ; Set the testable bit
       02C6'CF   10000001 8F  D0 0884                MOVL    #SS$_NORMAL!STS$M_INHIB_MSG,STATUS ; Set successful exit status
                                 088D                $EXIT_S STATUS                   ; Exit with the status
                                 0898
                                 0898       FAIL_OUT:                                 ; Failure exit
                        01    DD 0898                PUSHL   #1                                ; Arg count
                  00741132 8F  DD 089A                PUSHL   #UETP$_TEXT!STS$K_ERROR           ; Signal name
                        03    DD 08A0                PUSHL   #3                                ; Arg count
                      02DC    31 08A2                BRW     ERROR_EXIT                        ; Error exit
                                 08A5
```

```
                          08A5            .SBTTL  CHECKIOSB - Check IO status block
                          08A5       ;++
                          08A5       ; FUNCTIONAL DESCRIPTION:
                          08A5       ;        This routine checks the IO status block = #SS$_NORMAL
                          08A5       ;
                          08A5       ; CALLING SEQUENCE:
                          08A5       ;        CALLS   #1,CHECK_IOSB
                          08A5       ;
                          08A5       ; INPUT PARAMETERS:
                          08A5       ;        Address of error message
                          08A5       ;
                          08A5       ; IMPLICIT INPUTS:
                          08A5       ;        XD_IOSB is the IOSB from some $QIO
                          08A5       ;
                          08A5       ; OUTPUT PARAMETERS:
                          08A5       ;        NONE
                          08A5       ;
                          08A5       ; IMPLICIT OUTPUTS:
                          08A5       ;        Exit with status if IOSB not right
                          08A5       ;
                          08A5       ; COMPLETION CODES:
                          08A5       ;        IO status in STATUS if error
                          08A5       ;
                          08A5       ; SIDE EFFECTS:
                          08A5       ;        Program exit if error found
                          08A5       ;
                          08A5       ;--
                          08A5
                          08A5       CHECK_IOSB:
                   0004   08A5               .WORD   ^M<R2>
01     0670'CF      B1    08A7               CMPW    XD_IOSB,#SS$_NORMAL       ; Is the QIO O.K.?
            01      12    08AC               BNEQ    10$                       ; Br if not
                    04    08AE               RET                               ; Return
                          08AF       10$:
7E     0670'CF      3C    08AF               MOVZWL  XD_IOSB,-(SP)             ; Push the error status code
02C6'CF     6E      D0    08B4               MOVL    (SP),STATUS               ; Set return status
52     0411'CF      DE    08B9               MOVAL   DMP_IOSB_DUMP,R2          ; Assume we're testing a DMP
           00'      91    08BE               CMPB    S^#DT$_DMP11,-            ; But are we?
       0253'CF            08C0                       DIBBUF+DIB$B_DEVTYPE
            D5      13    08C3               BEQL    20$
52     0393'CF      DE    08C5               MOVAL   DMF_IOSB_DUMP,R2          ; Get a different string if not
                          08CA       20$:
                          08CA               $FAO_S  CTRSTR = (R2),-           ; Get the IOSB in plain text
                          08CA                       OUTLEN = BUFFER_PTR,-
                          08CA                       OUTBUF = FAO_BUF,-
                          08CA                       P1     = @#XD_IOSB,-
                          08CA                       P2     = @#XD_IOSB+2,-
                          08CA                       P3     = @#XD_IOSB+4,-
                          08CA                       P4     = @#XD_IOSB+5,-
                          08CA                       P5     = @#XD_IOSB+6,-
                          08CA                       P6     = @#XD_IOSB+7
       000C'CF      DF    08FF               PUSHAL  BUFFER_PTR
            01      DD    0903               PUSHL   #1
  00741132 8F       DD    0905               PUSHL   #UETP$_TEXT!STS$K_ERROR
         04 AC      DD    0908               PUSHL   04(AP)
            01      DD    090E               PUSHL   #1
  00741132 8F       DD    0910               PUSHL   #UETP$_TEXT!STS$K_ERROR
```

```
      07   DD  0916  1208         PUSHL   #7                        ; Argument count
    0266   31  0918  1209         BRW     ERROR_EXIT                ; Error exit
               091B  1210
```

```
                          091B    .SBTTL  Check QIO AST Routine
                          091B  ;++
                          091B  ; FUNCTIONAL DESCRIPTION:
                          091B  ;       This routine will be called as a QIO completion AST routine
                          091B  ;       It checks IO status block and the AST parameter
                          091B  ;
                          091B  ; CALLING SEQUENCE:
                          091B  ;       Called via AST at $QIO completion
                          091B  ;
                          091B  ; INPUT PARAMETERS:
                          091B  ;       NONE
                          091B  ;
                          091B  ; IMPLICIT INPUTS:
                          091B  ;       DEVDSC, ALT_BUFFER_PTR, ALT_FAO_BUF and ALT_BUFFER used in forming a
                          091B  ;               potential error message.
                          091B  ;
                          091B  ; OUTPUT PARAMETERS:
                          091B  ;       NONE
                          091B  ;
                          091B  ; IMPLICIT OUTPUTS:
                          091B  ;       Error message if error
                          091B  ;
                          091B  ; COMPLETION CODES:
                          091B  ;       IO status in STATUS if error
                          091B  ;
                          091B  ; SIDE EFFECTS:
                          091B  ;       Program exit if error
                          091B  ;       BUFFER_PTR and BUFFER used if error
                          091B  ;
                          091B  ;--
                          091B  CHK_QIO_AST:
                    0FFC  091B          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                          091D
                          091D          $FAO_S  CTRSTR = @04(AP),-             ; Form message for CHECK_IOSB
                          091D                  OUTLEN = ALT_BUFFER_PTR,-
                          091D                  OUTBUF = ALT_FAO_BUF,-
                          091D                  P1     = #DEVDSC
        0116'CF     DF    0935          PUSHAL  ALT_BUFFER_PTR
   FF67 CF     01   FB    0939          CALLS   #1,CHECK_IOSB                  ; Go check IO status block
                    04    093E          RET
```

```
                                  093F         .SBTTL  Receive data AST routine
                                  093F  ;++
                                  093F  ; FUNCTIONAL DESCRIPTION:
                                  093F  ;       This routine will be called as receive data AST routine
                                  093F  ;       It checks IO status and compare the data in the receive buffer
                                  093F  ;       against the transmit buffer
                                  093F  ;
                                  093F  ; CALLING SEQUENCE:
                                  093F  ;       Called via AST at $QIO READ
                                  093F  ;
                                  093F  ; INPUT PARAMETERS:
                                  093F  ;       AST parameter = message length
                                  093F  ;
                                  093F  ; IMPLICIT INPUTS:
                                  093F  ;       DEVDSC and various text buffers are used in forming error messages
                                  093F  ;
                                  093F  ; OUTPUT PARAMETERS:
                                  093F  ;       NONE
                                  093F  ;
                                  093F  ; IMPLICIT OUTPUTS:
                                  093F  ;       Error message if error found
                                  093F  ;
                                  093F  ; COMPLETION CODES:
                                  093F  ;       in STATUS
                                  093F  ;
                                  093F  ; SIDE EFFECTS:
                                  093F  ;       Program exit if error found
                                  093F  ;
                                  093F  ;--
                                  093F  RECV_AST:
                           0FFC   093F          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
            01   0678'CF     B1   0941          CMPW    RCV_IOSB,#SS$_NORMAL      ; Is the QIO O.K.?
                      0C     12   0946          BNEQ    10$                       ; Br if not
 0680'CF   0880'CF   04 AC   29   0948          CMPC3   4(AP),RECV_BUF,XMIT_BUF   ; Compare the data
                      2A     12   0951          BNEQ    20$
                             04   0953          RET
                                  0954  10$:
                                  0954          $FAO_S  CTRSTR = READ_PRM,-
                                  0954                  OUTLEN = ALT_BUFFER_PTR,-
                                  0954                  OUTBUF = ALT_FAO_BUF,-
                                  0954                  P1     = #DEVDSC
      0670'CF    0678'CF     7D   096D          MOVQ    RCV_IOSB,XD_IOSB          ; Set up a copy of our error status
                 0116'CF     DF   0974          PUSHAL  ALT_BUFFER_PTR
           FF28 CF   01      FB   0978          CALLS   #1,CHECK_IOSB             ; Take advantage of existing routine
                                  097D          ; Note that we will not return!
                                  097D
                                  097D  20$:
                7E   61      9A   097D          MOVZBL  (R1),-(SP)                ; Save the bad data...
                7E   63      9A   0980          MOVZBL  (R3),-(SP)                ; ...the good data...
           7E   04 AC    50  C3   0983          SUBL3   R0,4(AP),-(SP)            ; ...the offset of the mismatch...
             7E  02E2'CF     3C   0988          MOVZWL  UNIT_NUMBER,-(SP)         ; ...the failing unit...
                 0218'CF     7F   098D          PUSHAQ  DEVDSC                    ; ...the device name...
                      05     DD   0991          PUSHL   #5                        ; ...and the count of parameters...
             00748012 8F     DD   0993          PUSHL   #UETP$_DATAER!STS$K_ERROR ; ...for our error message
                      07     DD   0999          PUSHL   #7
                    01E3     31   099B          BRW     ERROR_EXIT
```

```
                        099E            .SBTTL  Half Minute Timer Expiration Routine
                        099E    ;++
                        099E    ; FUNCTIONAL DESCRIPTION:
                        099E    ;       This routine will be called only if the timer which was set to prevent
                        099E    ;       program hangs goes off.
                        099E    ;
                        099E    ; CALLING SEQUENCE:
                        099E    ;       Called via AST at $SETIMR expiration.
                        099E    ;
                        099E    ; INPUT PARAMETERS:
                        099E    ;       04(AP)  Address of a descriptor for an error message
                        099E    ;
                        099E    ; IMPLICIT INPUTS:
                        099E    ;       DEVDSC and various text buffers are used to for error messages
                        099E    ;
                        099E    ; OUTPUT PARAMETERS:
                        099E    ;       NONE
                        099E    ;
                        099E    ; IMPLICIT OUTPUTS:
                        099E    ;       Time out error message
                        099E    ;
                        099E    ; COMPLETION CODES:
                        099E    ;       NONE
                        099E    ;
                        099E    ; SIDE EFFECTS:
                        099E    ;       Program exit
                        099E    ;
                        099E    ;--
                        099E
                        099E    TIME_ERR_OUT:
                 0FFC   099E            .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        09A0
                        09A0            $FAO_S  CTRSTR = @04(AP),-
                        09A0                    OUTLEN = ALT_BUFFER_PTR,-
                        09A0                    OUTBUF = ALT_FAO_BUF,-
                        09A0                    P1     = #DEVDSC
       0116'CF     DF   09B8            PUSHAL  ALT_BUFFER_PTR          ; Set up our error message
            01     DD   09BC            PUSHL   #1
   00741132 8F     DD   09BE            PUSHL   #UETP$_TEXT!STS$K_ERROR
            03     DD   09C4            PUSHL   #3                      ; Push the argument count total
          01B8     31   09C6            BRW     ERROR_EXIT              ; Bail out completely
```

```
UETI
Syml

$$.
$$.
$$.
$$.
$$.
$$.
$$.
$$T
$$T
ACN
ALL
ALT
ALT
ALT
ARG
BAD
BEG
BEG
BUF
BUF
BUF
BUF
CCA
CHA
CHE
CHF
CHF
CHF
CHF
CHK
CLE
CNT
COM
COM
CON
CON
CS1
CS3
DDB
DEA
DEV
DEV
DEV
DEV
DEV
DIB
DIB
DIB
DIB
DIB
DMF
DMP
DTS
DUM
DUM
DVI
EFN
```

```
                          09C9           .SBTTL  Three Minutes Timer Expiration Routine
                          09C9      ;++
                          09C9      ; FUNCTIONAL DESCRIPTION:
                          09C9      ;        This routine will be called when the device test has been run for
                          09C9      ;        about three minutes. (that is, one normal run )
                          09C9      ;
                          09C9      ; CALLING SEQUENCE:
                          09C9      ;        Called via AST at $SETIMR expiration.
                          09C9      ;
                          09C9      ; INPUT PARAMETERS:
                          09C9      ;        NONE
                          09C9      ;
                          09C9      ; IMPLICIT INPUTS:
                          09C9      ;        NONE
                          09C9      ;
                          09C9      ; OUTPUT PARAMETERS:
                          09C9      ;        NONE
                          09C9      ;
                          09C9      ; IMPLICIT OUTPUTS:
                          09C9      ;        NONE
                          09C9      ;
                          09C9      ; COMPLETION CODES:
                          09C9      ;        NONE
                          09C9      ;
                          09C9      ; SIDE EFFECTS:
                          09C9      ;        Get out the transmit /receive loop test
                          09C9      ;
                          09C9      ;--
                          09C9
                          09C9      TIME_SUC_OUT:
                    0FFC  09C9           .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
    0002'CF    02     A8  09CB           BISW2   #TEST_OVERM,FLAG                   ; set test over bit
                      04  09D0           RET
```

```
09D1
09D1            .SBTTL  System Service Exception Handler
09D1     ;++
09D1     ; FUNCTIONAL DESCRIPTION:
09D1     ;       This routine is executed if a software or hardware exception occurs or
09D1     ;       if a LIB$SIGNAL system service is used to output a message.
09D1     ;
09D1     ; CALLING SEQUENCE:
09D1     ;       Entered via an exception from the system
09D1     ;
09D1     ; INPUT PARAMETERS:
09D1     ;       ERROR_COUNT    = previous cumulative error count
09D1     ;                        ------------------
09D1     ;          AP ---->      !        2       !
09D1     ;                        !----------------!
09D1     ;                        ! SIGNL ARY PNT  !
09D1     ;                        !----------------!
09D1     ;                        ! MECH  ARY PNT  !
09D1     ;                        !----------------!  ----------
09D1     ;                        !        4       !           ^
09D1     ;                        !----------------!           !
09D1     ;                        ! ESTABLISH FP   !           !
09D1     ;                        !----------------!           !
09D1     ;                        !      DEPTH     !  Mechanism Array
09D1     ;                        !----------------!           !
09D1     ;                        !       R0       !           !
09D1     ;                        !----------------!           !
09D1     ;                        !       R1       !           v
09D1     ;                        !----------------!  ----------
09D1     ;                        !        N       !           ^
09D1     ;                        !----------------!           !
09D1     ;                        ! CONDITION NAME !           !
09D1     ;                        !----------------!           !
09D1     ;                        ! N-3 ADDITIONAL !     Signal Array
09D1     ;                        ! LONG WORD ARGS !           !
09D1     ;                        !----------------!           !
09D1     ;                        !       PC       !           !
09D1     ;                        !----------------!           !
09D1     ;                        !       PSL      !           v
09D1     ;                        !----------------!  ----------
09D1     ; IMPLICIT INPUTS:
09D1     ;       NONE
09D1     ;
09D1     ; OUTPUT PARAMETERS:
09D1     ;       NONE
09D1     ;
09D1     ; IMPLICIT OUTPUTS:
09D1     ;       NONE
09D1     ;
09D1     ; COMPLETION CODES:
09D1     ;       SS$_NORMAL if it's a UETP condition or RMS error.
09D1     ;       Error status from exception, otherwise.
09D1     ;
09D1     ; SIDE EFFECTS:
09D1     ;       May branch to ERROR_EXIT.
09D1     ;       May print a message.
09D1     ;--
```

```
                         09D1   SSERROR:
                  0FFC   09D1           .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                         09D3
                         09D3           $SETAST_S ENBFLG = #0             ; Disable AST delivery
                 01  DD  09DC           PUSHL   #1                        ; Assume ASTs were enabled
            50   09  D1  09DE           CMPL    S^#SS$_WASSET,R0          ; Were ASTs enabled?
                 02  13  09E1           BEQL    10$                       ; BR if they were
                 6E  D4  09E3           CLRL    (SP)                      ; Set ASTs to remain disabled
                         09E5   10$:
                         09E5           $SETSFM_S ENBFLG = #0             ; Disable SS failure mode
                 01  DD  09EE           PUSHL   #1                        ; Assume SS failure mode was enabled
            50   09  D1  09F0           CMPL    S^#SS$_WASSET,R0          ; Was SS failure mode enabled?
                 02  13  09F3           BEQL    20$                       ; BR if it was
                 6E  D4  09F5           CLRL    (SP)                      ; Set SS failure mode to remain off
                         09F7   20$:
      56    04   AC  D0  09F7           MOVL    CHF$L_SIGARGLST(AP),R6    ; Get the signal array pointer
      59    04   A6  7D  09FB           MOVQ    CHF$L_SIG_NAME(R6),R9     ; Get NAME in R9 and ARG1 in R10
                 10  ED  09FF           CMPZV   #STS$V_FAC_NO,-           ; Is this a message from LIB$SIGNAL?
                 0C      0A01                   #STS$S_FAC_NO,-
 00000074   8F   59      0A02                   R9,#UETP$_FACILITY
                 14  12  0A08           BNEQ    30$                       ; BR if this is not a UETP exception
            66   02  C2  0A0A           SUBL2   #2,CHF$L_SIG_ARGS(R6)     ; Drop the PC and PSL
                         0A0D           $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6) ; Print the message
                 21  11  0A1C           BRB     40$                       ; Restore ASTs and SS fail mode
                         0A1E   30$:
 59    0000045C  8F  D1  0A1E           CMPL    #SS$_SSFAIL,R9            ; RMS failures are SysSvc failures
                 32  12  0A25           BNEQ    50$                       ; BR if this can't be an RMS failure
                 10  ED  0A27           CMPZV   #STS$V_FAC_NO,-           ; Is it an RMS failure?
                 0C      0A29                   #STS$S_FAC_NO,-
            01   5A      0A2A                   R10,#RMS$_FACILITY
                 2B  12  0A2C           BNEQ    50$                       ; BR if not
 5A    F0000000  8F  CA  0A2E           BICL2   #^XF0000000,R10           ; Strip control bits from status code
       08   A6   04  39  0A35           MATCHC  #4,CHF$L_SIG_ARG1(R6),-   ; Is it an RMS failure for which...
                 14      0A39                   #NRAT_LENGTH,-
          004D'CF        0A3A                   NO_RMS_AST_TABLE          ; ...no AST can be delivered?
                 1A  13  0A3D           BEQL    50$                       ; BR if so - must give error here
                         0A3F   40$:
                 01  BA  0A3F           POPR    #^M<R0>                   ; Restore SS failure mode...
                         0A41           $SETSFM_S ENBFLG = R0             ; ...
                 01  BA  0A4A           POPR    #^M<R0>                   ; Restore AST enable...
                         0A4C           $SETAST_S ENBFLG = R0             ; ...
            50   01  D0  0A55           MOVL    S^#SS$_NORMAL,R0          ; Supply a standard status for exit
                     04  0A58           RET                               ; Resume processing (or goto RMS_ERROR)
                         0A59   50$:
      02C6'CF    59  D0  0A59           MOVL    R9,STATUS                 ; Save the status
                 58  D4  0A5E           CLRL    R8                        ; Assume for now it's not SS failure
 59    0000045C  8F  D1  0A60           CMPL    #SS$_SSFAIL,R9            ; But is it a System Service failure?
                 38  12  0A67           BNEQ    70$                       ; BR if not - no special case message
                         0A69           $GETMSG_S MSGID = R10,-           ; Get SS failure code associated text
                         0A69                   MSGLEN = BUFFER_PTR,-
                         0A69                   BUFADR = FAO_BUF,-
                         0A69                   FLAGS  = #14,-
                         0A69                   OUTADR = MSG_BLOCK
        02EF'CF      95  0A80           TSTB    MSG_BLOCK+1               ; Get FAO arg count for SS failure code
                 16  13  0A84           BEQL    60$                       ; Don't use $GETMSG if no $FAO args...
        000C'CF      DF  0A86           PUSHAL  BUFFER_PTR                ; ...else build up...
```

```
UETI
Pse

PSE
---
. (
$AB
ROD
RWD
$RM
DMP

Pha
---
Ini
Com
Pas
Sym
Pas
Sym
Pse
Cro
Ass

The
171
The
181
59 (

Mac
---
_$2
_$2
_$2
TOT

265

The

MAC
```

```
                   01  DD  0A8A            PUSHL   #1                              ; ...a message describing...
          00741130 8F  DD  0A8C            PUSHL   #UETP$_TEXT                     ; ...why the System Service failed
                00 5A  F0  0A92            INSV    R10,#STS$V_SEVERITY,-           ; Give the message...
                6E 03      0A95                    #STS$S_SEVERITY,(SP)            ; ...the correct severity code
                58 03  D0  0A97            MOVL    #3,R8                           ; Count the number of args we pushed
                   05  11  0A9A            BRB     70$
                           0A9C       60$:
                   5A  DD  0A9C            PUSHL   R10                             ; Save SS failure code
                58 01  D0  0A9E            MOVL    #1,R8                           ; Count the number of args we pushed
                           0AA1       70$:
          57    66 04  C5  0AA1            MULL3   #4,CHF$L_SIG_ARGS(R6),R7 ; Convert longwords to bytes
                5E 57  C2  0AA5            SUBL2   R7,SP                            ; Save the current signal array...
       6E    04 A6 57  28  0AA8            MOVC3   R7,CHF$L_SIG_NAME(R6),(SP) ; ...on the stack
          7E    66 58  C1  0AAD            ADDL3   R8,CHF$L_SIG_ARGS(R6),-(SP) ; Push the current arg count
                 00CD  31  0AB1            BRW     ERROR_EXIT
```

```
                          0AB4         .SBTTL  RMS Error Handler
                          0AB4  ;++
                          0AB4  ; FUNCTIONAL DESCRIPTION:
                          0AB4  ;       This routine handles error returns from RMS calls.
                          0AB4  ;
                          0AB4  ; CALLING SEQUENCE:
                          0AB4  ;       Called by RMS when a file processing error is found.
                          0AB4  ;
                          0AB4  ; INPUT PARAMETERS:
                          0AB4  ;       The FAB or RAB associated with the RMS call.
                          0AB4  ;
                          0AB4  ; IMPLICIT INPUTS:
                          0AB4  ;       NONE
                          0AB4  ;
                          0AB4  ; OUTPUT PARAMETERS:
                          0AB4  ;       NONE
                          0AB4  ;
                          0AB4  ; IMPLICIT OUTPUTS:
                          0AB4  ;       Error message
                          0AB4  ;
                          0AB4  ; COMPLETION CODES:
                          0AB4  ;       NONE
                          0AB4  ;
                          0AB4  ; SIDE EFFECTS:
                          0AB4  ;       Program may exit, depending on severity of the error.
                          0AB4  ;
                          0AB4  ;--
                          0AB4
                          0AB4  RMS_ERROR:
                    0FFC  0AB4         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                          0AB6
       56    04 AC   D0   0AB6         MOVL    4(AP),R6                     ; See whether we're dealing with...
          66    03   91   0ABA         CMPB    #FAB$C_BID,FAB$B_BID(R6)  ; ...a FAB or a RAB
                16   12   0ABD         BNEQ    10$                          ; BR if it's a RAB
       57  01FD'CF   DE   0ABF         MOVAL   FILE,R7                      ; FAB-specific code:  text string...
          58    56   D0   0AC4         MOVL    R6,R8                        ; ...address of FAB...
             0C A6   DD   0AC7         PUSHL   FAB$L_STV(R6)                ; ...STV field for error...
             08 A6   DD   0ACA         PUSHL   FAB$L_STS(R6)                ; ...STS field for error...
   02C6'CF   08 A6   D0   0ACD         MOVL    FAB$L_STS(R6),STATUS         ; ...and save the error code
                15   11   0AD3         BRB     COMMON                       ; FAB and RAB share other code
                          0AD5  10$:
       57  0209'CF   DE   0AD5         MOVAL   RECORD,R7                    ; RAB-specific code:  text string...
       58    3C A6   D0   0ADA         MOVL    RAB$L_FAB(R6),R8             ; ...address of associated FAB...
             0C A6   DD   0ADE         PUSHL   RAB$L_STV(R6)                ; ...STV field for error...
             08 A6   DD   0AE1         PUSHL   RAB$L_STS(R6)                ; ...STS field for error...
   02C6'CF   08 A6   D0   0AE4         MOVL    RAB$L_STS(R6),STATUS         ; ...and save the error code
                          0AEA  COMMON:
       5A    34 A8   9A   0AEA         MOVZBL  FAB$B_FNS(R8),R10            ; Get the file name size
                          0AEE         $FAO_S  CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
                          0AEE                 OUTLEN = BUFFER_PTR,-
                          0AEE                 OUTBUF = FAO_BUF,-
                          0AEE                 P1     = R7,-
                          0AEE                 P2     = R10,-
                          0AEE                 P3     = FAB$L_FNA(R8)
        000C'CF      DF   0B08         PUSHAL  BUFFER_PTR                   ; ...and arguments for ERROR_EXIT...
                01   DD   0B0C         PUSHL   #1                           ; ...
   00741130 8F       DD   0B0E         PUSHL   #UETP$_TEXT                  ; ...
```

```
                00   EF  0B14  1573        EXTZV   #STS$V_SEVERITY,-
                03       0B16  1574                #STS$S_SEVERITY,-
       59   02C6'CF      0B17  1575                STATUS,R9               ; ...get the severity code...
            6E   59  88  0B1B  1576        BISB2   R9,(SP)                 ; ...and add it into the signal name
                 05  DD  0B1E  1577        PUSHL   #5                      ; Current arg count
              005E   31  0B20  1578        BRW     ERROR_EXIT
```

```
                                  0B23          .SBTTL  CTRL/C Handler
                                  0B23   ;++
                                  0B23   ; FUNCTIONAL DESCRIPTION:
                                  0B23   ;       This routine handles CTRL/C AST's
                                  0B23   ;
                                  0B23   ; CALLING SEQUENCE:
                                  0B23   ;       Called via AST
                                  0B23   ;
                                  0B23   ; INPUT PARAMETERS:
                                  0B23   ;       NONE
                                  0B23   ;
                                  0B23   ; IMPLICIT INPUTS:
                                  0B23   ;       NONE
                                  0B23   ;
                                  0B23   ; OUTPUT PARAMETERS:
                                  0B23   ;       NONE
                                  0B23   ;
                                  0B23   ; IMPLICIT OUTPUTS:
                                  0B23   ;       NONE
                                  0B23   ;
                                  0B23   ; COMPLETION CODES:
                                  0B23   ;       NONE
                                  0B23   ;
                                  0B23   ; SIDE EFFECTS:
                                  0B23   ;       NONE
                                  0B23   ;
                                  0B23   ;--
                                  0B23
                                  0B23   CCASTHAND:
                           0FFC   0B23          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                  0B25
        21 0002'CF    05     E1   0B25          BBC     #FLAG_SHUTDNV,FLAG,10$  ; Have to shut down device?
                                  0B2B          $QIO_S -                        ; Shut down the device
                                  0B2B                  CHAN = XD_CHAN,-
                                  0B2B                  FUNC = #IO$_SETMODE!IO$M_CTRL!IO$M_SHUTDOWN,-
                                  0B2B                  IOSB = XD_IOSB
                                  0B4C   10$:
              00A3'CF        DF   0B4C          PUSHAL  CNTRLCMSG               ; Set message pointer
                   01        DD   0B50          PUSHL   #1                      ; Set arg count
           00741130 8F       DD   0B52          PUSHL   #UETP$_TEXT!STS$K_WARNING ; Set signal name
                   00        DD   0B58          PUSHL   #0                      ; Indicate an abnormal termination
              0220'CF        DF   0B5A          PUSHAL  PROCESS_NAME            ; ...
                   02        DD   0B5E          PUSHL   #2                      ; ...
           007410E0 8F       DD   0B60          PUSHL   #UETP$_ABENDD!STS$K_WARNING ; ...
        00000000'GF   07     FB   0B66          CALLS   #7,G^LIB$SIGNAL         ; Output the message
                             D0   0B6D          MOVL    #<STS$M_INHIB_MSG!-     ; Set the exit status
                                  0B6E                  SS$_CONTROLC-=
                                  0B6E                  STS$K_SUCCESS+STS$K_WARNING>,-
    02C6'CF    10000650 8F        0B6E                  STATUS
                                  0B76          $EXIT_S STATUS                  ; Terminate program cleanly
                                  0B81             .SBTTL  Error Exit
                                  0B81       ;++
                                  0B81       ; FUNCTIONAL DESCRIPTION:
                                  0B81       ;         This routine prints an error message and exits.
                                  0B81       ;
                                  0B81       ; CALLING SEQUENCE:
                                  0B81       ;         MOVx  error status value,STATUS
                                  0B81       ;         PUSHx error specific information on the stack
                                  0B81       ;         PUSHL current argument count
                                  0B81       ;         BRW   ERROR_EXIT
                                  0B81       ;
                                  0B81       ; INPUT PARAMETERS:
                                  0B81       ;         Arguments to LIB$SIGNAL, as above
                                  0B81       ;
                                  0B81       ; IMPLICIT INPUTS:
                                  0B81       ;         NONE
                                  0B81       ;
                                  0B81       ; OUTPUT PARAMETERS:
                                  0B81       ;         Message to SYS$OUTPUT and SYS$ERROR
                                  0B81       ;
                                  0B81       ; IMPLICIT OUTPUTS:
                                  0B81       ;         Program exit
                                  0B81       ;
                                  0B81       ; COMPLETION CODES:
                                  0B81       ;         Error in STATUS
                                  0B81       ;
                                  0B81       ; SIDE EFFECTS:
                                  0B81       ;         NONE
                                  0B81       ;
                                  0B81       ;--
                                  0B81
                                  0B81       ERROR_EXIT:
                                  0B81
                                  0B81               $SETAST_S ENBFLG = #0          ; ASTs can play havoc with messages
     15 0002'CF    03     E0      0B8A               BBS     #BEGIN_MSGV,FLAG,10$     ; BR if "begin" msg already printed
                   7E     D4      0B90               CLRL    -(SP)                    ; Set the time stamp flag
              000F'CF     DF      0B92               PUSHAL  TEST_NAME                ; Set the test name
                   02     DD      0B96               PUSHL   #2                       ; Push the argument count
          00741039 8F     DD      0B98               PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
       00000000'GF        04 FB   0B9E               CALLS   #4,G^LIB$SIGNAL          ; Print the startup message
                                  0BA5       10$:
    0302'CF    08  8E     C1      0BA5               ADDL3   (SP)+,#8,ARG_COUNT       ; Get total # args, pop partial count
              02C2'CF     D6      0BAB               INCL    ERROR_COUNT              ; Keep running error count
                   00     DD      0BAF               PUSHL   #0                       ; Push the time parameter
              0220'CF     DF      0BB1               PUSHAL  PROCESS_NAME             ; Push test name...
          000F0002 8F     DD      0BB5               PUSHL   #^XF0002                 ; ...arg count...
          007410E2 8F     DD      0BBB               PUSHL   #UETP$_ABENDD!STS$K_ERROR ; ...and signal name
              02C2'CF     DD      0BC1               PUSHL   ERROR_COUNT              ; Finish off arg list...
              0220'CF     DF      0BC5               PUSHAL  PROCESS_NAME             ; ...
          00010002 8F     DD      0BC9               PUSHL   #^X10002                 ; ...
          00748022 8F     DD      0BCF               PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
   00000000'GF   0302'CF  FB      0BD5               CALLS   ARG_COUNT,G^LIB$SIGNAL   ; Truly bitch
                                  0BDE
              02C6'CF     D5      0BDE               TSTL    STATUS                   ; Did we exit with an error code?
                   09     12      0BE2               BNEQ    20$                      ; BR if we did
          007410E2 8F     D0      0BE4               MOVL    #UETP$_ABENDD!STS$K_ERROR,- ; Supply a generic one otherwise
              02C6'CF             0BEA                       STATUS
```

```
                                    0BED  20$:
        1F 0002'CF    05    E1      0BED          BBC      #FLAG_SHUTDNV,FLAG,30$  ; Have to shut down device?
                                    0BF3          $QIO_S -                         ; Shut down the device
                                    0BF3                   CHAN = XD_CHAN,-
                                    0BF3                   FUNC = #IO$_SETMODE!IO$M_CTRL!IO$M_SHUTDOWN
                                    0C12  30$:
   02C6'CF    10000000 8F   C8      0C12          BISL     #STS$M_INHIB_MSG,STATUS ; Don't print messages twice!
                                    0C1B          $EXIT_S  STATUS                  ; Exit in error
```

```
                              0C26          .SBTTL  Exit Handler
                              0C26  ;++
                              0C26  ; FUNCTIONAL DESCRIPTION:
                              0C26  ;       This routine handles cleanup at exit.  If the MODE logical name is
                              0C26  ;       equated to "ONE", the routine will update the test flag in the
                              0C26  ;       UETINIDEV.DAT file depending on the UETUNT$M_TESTABLE flag state in the
                              0C26  ;       UETUNT$B_FLAGS field of the unit block for each unit for the device
                              0C26  ;       under test.
                              0C26  ;
                              0C26  ; CALLING SEQUENCE:
                              0C26  ;       Invoked automatically by $EXIT System Service.
                              0C26  ;
                              0C26  ; INPUT PARAMETERS:
                              0C26  ;       STATUS  contains the exit status.
                              0C26  ;       FLAG    has synchronizing bits.
                              0C26  ;       DDB_RFA contains the RFA of the DDB record for this device in UETINIDEV.
                              0C26  ;
                              0C26  ; IMPLICIT INPUTS:
                              0C26  ;       UNIT_LIST points to the head of a doubly linked circular list of unit
                              0C26  ;                 blocks for the device under test.
                              0C26  ;
                              0C26  ; OUTPUT PARAMETERS:
                              0C26  ;       NONE
                              0C26  ;
                              0C26  ; IMPLICIT OUTPUTS:
                              0C26  ;       Various files are de-accessed, the process name is reset, and any
                              0C26  ;       necessary synchronization with UETPDEV01 is carried out.
                              0C26  ;       If the MODE logical name is equated to "ONE", the routine will update
                              0C26  ;       the test flag in the UETINIDEV.DAT file depending on the
                              0C26  ;       UETUNT$M_TESTABLE flag state in the UETUNT$B_FLAGS field of the unit
                              0C26  ;       block for each unit for the device under test.
                              0C26  ;
                              0C26  ; COMPLETION CODES:
                              0C26  ;       NONE
                              0C26  ;
                              0C26  ; SIDE EFFECTS:
                              0C26  ;       NONE
                              0C26  ;
                              0C26  ;--
                              0C26
                              0C26  EXIT_HANDLER:
                        0FFC  0C26          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                              0C28
                              0C28          $SETSFM_S ENBFLG = #0           ; Turn off System Service failure mode
                              0C31          $SETAST_S ENBFLG = #0           ; No more ASTs
                              0C3A          $TRNLOG_S LOGNAM = MODE,-       ; Get the run mode
                              0C3A                  RSLLEN = BUFFER_PTR,-
                              0C3A                  RSLBUF = FAO_BUF
     0014'CF     20     8A    0C53          BICB2   #LC_BITM,BUFFER         ; Convert to upper case
  0014'CF     4F 8F     91    0C58          CMPB    #^A?O/,BUFFER           ; Is this a one shot?
                 03     13    0C5E          BEQL    10$                     ; BR if yes...
               00B8     31    0C60          BRW     END_UPDATE              ; ...else don't update UETINIDEV.DAT
                              0C63  10$:
  03 0002'CF     02     E0    0C63          BBS     #SAFE_TO_UPDV,FLAG,20$  ; Only update if it's safe
               00AF     31    0C69          BRW     END_UPDATE              ; Else forget it
                              0C6C  20$:
    5A   0B7C'CF        DE    0C6C          MOVAL   INI_RAB,R10             ; Set the RAB address
```

```
            1E AA    02   90  0C71               MOVB    #RAB$C_RFA,RAB$B_RAC(R10)  ; Set RFA mode
   10 AA  0BC0'CF    06   28  0C75               MOVC3   #6,DDB_RFA,RAB$W_RFA(R10)  ; Set RFA to DDB line
                              0C7C               $GET    RAB = (R10)                ; Go back to the DDB record
               75 50      E9  0C85               BLBC    R0,UPDATE_FAILED           ; If failure then forget it
            1E AA    00   90  0C88               MOVB    #RAB$C_SEQ,RAB$B_RAC(R10)  ; Set back to sequential mode
 5B  0A88'CF  00000A88'8F C1  0C8C               ADDL3   #UNIT_LIST,UNIT_LIST,R11   ; Set the unit block list header
                     59   D4  0C96               CLRL    R9                         ; Init a counter
                              0C98       UNIT_LOOP:
                     01   E1  0C98               BBC     #UETUNT$V_TESTABLE,-       ; BR if this unit is not testable
               02 0B AB       0C9A                       UETUNT$B_FLAGS(R11),10$
                     59   D6  0C9D               INCL    R9                         ; Count testable units
                              0C9F       10$:
               5B    6B   C0  0C9F               ADDL2   (R11),R11                  ; Next unit block
        00000A88'8F  5B   D1  0CA2               CMPL    R11,#UNIT_LIST             ; Are we full circle in the list?
                     ED   12  0CA9               BNEQ    UNIT_LOOP                  ; BR if not
                     59   D5  0CAB               TSTL    R9                         ; Any testable units?
                     12   12  0CAD               BNEQ    20$                        ; BR if yes...
         0018'CF  4E 8F   90  0CAF               MOVB    #^A/N/,BUFFER+4            ; ...else disable the DDB record...
                              0CB5               $UPDATE RAB = (R10)                ; ...here
               3C 50      E9  0CBE               BLBC    R0,UPDATE_FAILED           ; If error then forget it
                              0CC1       20$:
               5B    6B   C0  0CC1               ADDL2   (R11),R11                  ; Next unit block
        00000A88'8F  5B   D1  0CC4               CMPL    R11,#UNIT_LIST             ; Are we full circle in the list?
                     4E   13  0CCB               BEQL    END_UPDATE                 ; BR if yes
                              0CCD               $GET    RAB = (R10)                ; Get a record
               24 50      E9  0CD6               BLBC    R0,UPDATE_FAILED           ; If error then forget it
            0014'CF  20   8A  0CD9               BICB2   #LC_BITM,BUFFER            ; Convert to uppercase
         0014'CF  55 8F   91  0CDE               CMPB    #^A7U/,BUFFER              ; Is it a UCB record?
                     35   12  0CE4               BNEQ    END_UPDATE                 ; BR if not
                     01   E0  0CE6               BBS     #UETUNT$V_TESTABLE,-       ; BR if this unit is testable...
               D6 0B AB       0CE8                       UETUNT$B_FLAGS(R11),20$
         0018'CF  4E 8F   90  0CEB               MOVB    #^A/N/,BUFFER+4            ; ...else disable the UCB record...
                              0CF1               $UPDATE RAB = (R10)                ; ...here
               C4 50      E8  0CFA               BLBS    R0,20$                     ; Look at the next record if no error
                              0CFD       UPDATE_FAILED:
               0C AA      DD  0CFD               PUSHL   RAB$L_STV(R10)             ; Do a simple message...
                     50   DD  0D00               PUSHL   R0                         ; ...to tell of the failure
                01B8'CF   DF  0D02               PUSHAL  INIDEV_UPDERR
                     01   DD  0D06               PUSHL   #1
                     00   EF  0D08               EXTZV   #STS$V_SEVERITY,-          ; Copy the severity from RMS status...
          7E    50   03       0D0A                       #STS$S_SEVERITY,R0,-(SP)
   6E   00741130 8F       C5  0D0D               BISL2   #UETP$_TEXT,(SP)           ; ...to our message
   00000000'GF       05   FB  0D14               CALLS   #5,G^LIB$SIGNAL
                              0D1B       END_UPDATE:
                     00   DD  0D1B               PUSHL   #0                         ; Set the time flag
                000F'CF   DF  0D1D               PUSHAL  TEST_NAME                  ; Push the test name
                     02   DD  0D21               PUSHL   #2                         ; Push arg count
                     00   EF  0D23               EXTZV   #STS$V_SEVERITY,-          ; Push the proper exit severity...
                     03       0D25                       #STS$S_SEVERITY,-
          7E   02C6'CF        0D26                       STATUS,-(SP)
   6E   00741080 8F       C8  0D2A               BISL2   #UETP$_ENDEDD,(SP)         ; ...and use it in our message code
                     04   DD  0D31               PUSHL   #4
               51    5E   D0  0D33               MOVL    SP,R1
                              0D36               $PUTMSG_S MSGVEC = (R1)            ; Output the message
                              0D45               $SETPRN_S PRCNAM = ACNT_NAME       ; Reset the process name
                          04  0D50               RET                                ; That's all folks!
                              0D51
```

```
                                      0D51  1811          .END     UETDMPF00
```

```
                                                                  G  4
UETDMPF00                                VAX/VMS UETP DEVICE TEST FOR DMP 11/ DMF 16-SEP-1984 01:24:05  VAX/VMS Macro V04-00         Page  46
Symbol table                                                                      10-SEP-1984 12:03:55  [UETP.SRC]UETDMPF00.MAR;2        (22)

$$.TAB                          = 00000C68 R      03    END_UPDATE                        00000D18 R      05
$$.TABEND                       = 00000CAC R      03    ERRCNT_BUF                        00000470 R      03
$$.TMP                          = 00000000              ERRCNT_LEN                      = 00000200
$$.TMP1                         = 00000001              ERRCOUNT_DESC                     00000468 R      03
$$.TMP2                         = 0000006A              ERROR_COUNT                       000002C2 R      03
$$.TMPX                         = 00000016 R      04    ERROR_EXIT                        00000881 R      05
$$.TMPX1                        = 0000000D              ERROR_TEST                        00000663 R      05
$$T1                            = 00000001              ERRTEST_MSG                       000005C0 R      02
$$T2                            = 00000006              ERRTST_ERR                        000006DF R      05
ACNT_NAME                         00000000 R      02    ERRTST_P2BUF                      00000460 R      03
ALL_SET                           000003E6 R      05    ERRTST_P2DESC                     00000458 R      03
ALT_BUFFER                        0000011E R      03    ERRTST_P2LEN                    = 00000008
ALT_BUFFER_PTR                    00000116 R      03    ESC                             = 0000001B
ALT_FAO_BUF                       0000010E R      03    EXIT_DESC                         000002F2 R      03
ARG_COUNT                         00000302 R      03    EXIT_HANDLER                      00000C26 R      05
BAD_DATA                          00000A80 R      03    FAB$B_BID                       = 00000000
BEGIN_MSGM                      = 00000008              FAB$B_FNS                       = 00000034
BEGIN_MSGV                      = 00000003              FAB$C_BID                       = 00000003
BUFFER                            00000014 R      03    FAB$C_BLN                       = 00000050
BUFFER_PTR                        0000000C R      03    FAB$C_SEQ                       = 00000000
BUF_DESC                          0000030A R      03    FAB$C_VAR                       = 00000002
BUF_LEN                           00000308 R      03    FAB$L_ALQ                       = 00000010
CCASTHAND                         00000B23 R      05    FAB$L_DEV                       = 00000040
CHAN_BUF                          00000312 R      03    FAB$L_FNA                       = 0000002C
CHECK_IOSB                        000008A5 R      05    FAB$L_FOP                       = 00000004
CHF$L_SIGARGLST                 = 00000004              FAB$L_STS                       = 00000008
CHF$L_SIG_ARG1                  = 00000008              FAB$L_STV                       = 0000000C
CHF$L_SIG_ARGS                  = 00000000              FAB$V_CHAN_MODE                 = 00000002
CHF$L_SIG_NAME                  = 00000004              FAB$V_CR                        = 00000001
CHK_QIO_AST                       0000091B R      05    FAB$V_FILE_MODE                 = 00000004
CLEAN_EXIT                        000007C8 R      05    FAB$V_GET                       = 00000001
CNTRLCMSG                         000000A3 R      02    FAB$V_LNM_MODE                  = 00000000
COMMON                            00000AEA R      05    FAB$V_PUT                       = 00000000
COMP_STATUS_MSG                   000004AF R      02    FAB$V_UFO                       = 00000011
CONTROLLER                        00000031 R      02    FAB$V_UPD                       = 00000003
CONT_DESC                         000001F5 R      02    FAB$V_UPI                       = 00000006
CS1                               00000082 R      02    FAB$W_GBC                       = 00000048
CS3                               00000094 R      02    FAIL_OUT                          00000898 R      05
DDB_RFA                           00000BC0 R      03    FAO_BUF                           00000004 R      03
DEAD_CTRLNAME                     000000E4 R      02    FILE                              000001FD R      02
DEV$V_TRM                       = 00000002              FIND_IT                           000001E1 R      05
DEVDEP_SIZE                     = 00000000              FLAG                              00000002 R      03
DEVDSC                            00000218 R      03    FLAG_SHUTDNM                    = 00000020
DEVNAM_LEN                        000002E4 R      03    FLAG_SHUTDNV                    = 00000005
DEV_NAME                          00000237 R      03    FOUND_IT                          00000279 R      05
DIB                               00000246 R      03    GOOD_DATA                         00000A81 R      03
DIB$B_DEVCLASS                  = 00000004              HALFMIN                           000001E5 R      02
DIB$B_DEVTYPE                   = 00000005              ILLEGAL_REC                       00000151 R      02
DIB$K_LENGTH                    = 00000074              INADDRESS                         000002D2 R      03
DIBBUF                            0000024E R      03    INIDEV_UPDERR                     000001B8 R      02
DMF_IOSB_DUMP                     00000393 R      02    INI_FAB                           00000B2C R      03
DMP_IOSB_DUMP                     00000411 R      02    INI_RAB                           00000B7C R      03
DTS_DMP1T                         ********   X    05    INPUT_ITMLST                      00000072 R      02
DUMMY_FAB                         00000C18 R      03    IO$M_CLR_COUNT                    ********   X    05
DUMMY_RAB                         00000C68 R      03    IO$M_CTRL                         ********   X    05
DVI$_DEVNAM                     = 00000020              IO$M_CTRLCAST                     ********   X    05
EFN2                            = 00000004              IO$M_NOW                          ********   X    05
```

```
                                                                   H  4
UETDMPF00                          VAX/VMS UETP DEVICE TEST FOR DMP 11/ DMF 16-SEP-1984 01:24:05  VAX/VMS Macro V04-00        Page  47
Symbol table                                                                10-SEP-1984 12:03:55  [UETP.SRC]UETDMPF00.MAR;2       (22)

IO$M_RD_COUNT                     ********    X    05          RAB$L_STS                     = 00000008
IO$M_SHOTDOWN                     ********    X    05          RAB$L_STV                     = 0000000C
IO$M_STARTUP                      ********    X    05          RAB$V_PMT                     = 0000001E
IO$_READVBLK                      ********    X    05          RAB$W_RFA                     = 00000010
IO$_SENSEMODE                     ********    X    05          RAB$W_RSZ                     = 00000022
IO$_SETMODE                       ********    X    05          RCV_IOSB                        00000678 R        03
IO$_WRITEVBLK                     ********    X    05          READ_ERRCOUNT                   00000778 R        05
ITERATION                         000002E6 R       03          READ_PRM                        0000031A R        02
LC_BITM                         = 00000020                     READ_SIZE                     = 00000000
LIB$SIGNAL                        ********    X    05          RECEIVED_MSG                    000004E7 R        02
LIMIT                           = 00000010                     RECORD                          00000209 R        02
LOOPBACK_TEST                     0000051D R       05          RECV                            00000581 R        05
MAX_DEV_DESIG                   = 0000000A                     RECVPOOL_SIZ                  = 00000004
MAX_MSG_LEN                     = 00000200                     RECV_AST                        0000093F R        05
MAX_PROC_NAME                   = 0000000F                     RECV_BUF                        00000880 R        03
MAX_UNIT_DESIG                  = 00000005                     RECV_EFN                      = 00000008
MODE                              00000041 R       02          RECV_ERR_MSG                    000004FF R        02
MODE_IS_ONEM                    = 00000010                     REC_SIZE                      = 00000028
MODE_IS_ONEV                    = 00000004                     RESTART                         0000043D R        05
MSG_BLOCK                         000002EE R       03          RMS$_BLN                        ********    X     02
NAME_LEN                        = 0000000F                     RMS$_BUSY                       ********    X     02
NEW_NODE                          00000A90 R       03          RMS$_CDA                        ********    X     02
NMA$C_LINCN_LOO                 = 00000001                     RMS$_FAB                        ********    X     02
NMA$C_LINPR_POI                 = 00000000                     RMS$_FACILITY                 = 00000001
NMA$C_PCCI_TRI                  = 00000474                     RMS$_RAB                        ********    X     02
NMA$C_PCLI_CON                  = 00000456                     RMS_ERROR                       00000AB4 R        05
NMA$C_PCLI_PRO                  = 00000458                     RMS_ERR_STRING                  00000217 R        02
NOUNIT_SELECTED                   0000012B R       02          RW_TIME_ID                    = 00000003
NO_CTRLNAME                       000000C4 R       02          RW_TO_MSG                       00000275 R        02
NO_RMS_AST_TABLE                  0000004D R       02          SAFE_TO_UPDM                  = 00000004
NRAT_LENGTH                     = 00000014                     SAFE_TO_UPDV                  = 00000002
OTS$CVT_TI_L                      ********    X    05          SEC$M_EXPREG                    ********    X     05
OUTADDRESS                        000002DA R       03          SEC$M_GBL                       ********    X     05
P1BUF                             00000386 R       03          SENSE_ERRMSG                    00000541 R        02
P2BUF                             0000039E R       03          SENSE_P1BUF                     000003B8 R        03
P2BUF_DESC                        00000396 R       03          SENSE_P2BUF                     000003C8 R        03
P2BUF_LEN                       = 0000000C                     SENSE_P2DESC                    000003C0 R        03
PAGES                           = 00000001                     SENSE_P2LEN                   = 00000090
PASS                              000002EA R       03          SENSE_PRM                       00000339 R        02
PASS_MSG                          00000185 R       02          SENSE_TEST                      000005D6 R        05
PMTSIZ                          = 00000019                     SET_PRM                         00000366 R        02
PRM                             = 00000064                     SET_XMIT_BUF                    0000052B R        05
PROCESS_NAME                      00000220 R       03          SHR$_ABENDD                   = 000010E0
PROCESS_NAME_FREE               = 0000000B                     SHR$_BEGIND                   = 00001038
PROC_CONT_NAME                    0000008B R       05          SHR$_ENDEDD                   = 00001080
PROMPT                            00000238 R       02          SHR$_OPENIN                   = 00001098
QUAD_STATUS                       000002CA R       03          SHR$_TEXT                     = 00001130
RAB$B_PSZ                       = 00000034                     SS$_BADPARAM                  = 00000014
RAB$B_RAC                       = 0000001E                     SS$_BUFFEROVF                 = 00000601
RAB$C_BID                       = 00000001                     SS$_CONTROLC                  = 00000651
RAB$C_BLN                       = 00000044                     SS$_ENDOFFILE                 = 00000870
RAB$C_RFA                       = 00000002                     SS$_NORMAL                    = 00000001
RAB$C_SEQ                       = 00000000                     SS$_NOSUCHSEC                 = 00000978
RAB$L_CTX                       = 00000018                     SS$_SSFAIL                    = 0000045C
RAB$L_FAB                       = 0000003C                     SS$_WASSET                    = 00000009
RAB$L_PBF                       = 00000030                     SSERROR                         000009D1 R        05
RAB$L_ROP                       = 00000004                     SS_SYNCH_EFN                  = 00000003
```

```
                                                                         I 4
UETDMPF00                       VAX/VMS UETP DEVICE TEST FOR DMP 11/ DMF 16-SEP-1984 01:24:05  VAX/VMS Macro V04-00          Page  48
Symbol table                                                             10-SEP-1984 12:03:55  [UETP.SRC]UETDMPF00.MAR;2          (22)

START_CONT                        00000461 R      05        TR_P2BUF_DESC                    000003AA R      03
START_CONT_PRM                    000002A2 R      02        TR_P2BUF_LEN                   = 00000006
START_TEST                        00000407 R      05        TTCHAN                           00000000 R      03
START_TO_MSG                      00000251 R      02        UETDMPF00                        00000000 RG     05
START_TRI                         00000492 R      05        UETP                           = 00740000
START_TRIB_PRM                    000002D0 R      02        UETP$_ABENDD                   = 007410E0
STATUS                            000002C6 R      03        UETP$_ABORTC                   = 0074832B
STR$UPCASE                        ********    X   05        UETP$_BEGIND                   = 00741038
STS$K_ERROR                     = 00000002                  UETP$_DATAER                   = 00748010
STS$K_INFO                      = 00000003                  UETP$_DENOSU                   = 00748333
STS$K_SUCCESS                   = 00000001                  UETP$_DEUNUS                   = 0074819A
STS$K_WARNING                   = 00000000                  UETP$_ENDEDD                   = 00741080
STS$M_INHIB_MSG                 = 10000000                  UETP$_ERBOXPROC                = 00748020
STS$S_FAC_NO                    = 0000000C                  UETP$_FACILITY                 = 00000074
STS$S_SEVERITY                  = 00000003                  UETP$_OPENIN                   = 00741098
STS$V_FAC_NO                    = 00000010                  UETP$_TEXT                     = 00741130
STS$V_SEVERITY                  = 00000000                  UETUNT$B_FLAGS                 = 0000000B
SUC_EXIT                          00000810 R      05        UETUNT$B_TYPE                  = 00000008
SUPDEV_GBLSEC                     00000020 R      02        UETUNT$C_FAB                   = 00000110
SUP_FAB                           00000BCB R      03        UETUNT$C_INDSIZ                = 000001A4
SYS$ASSIGN                        ********   GX   05        UETUNT$K_FAB                   = 00000110
SYS$CANTIM                        ********   GX   05        UETUNT$K_RAB                   = 00000160
SYS$CONNECT                       ********   GX   05        UETUNT$M_TESTABLE              = 00000002
SYS$CRMPSC                        ********   GX   05        UETUNT$T_FILSPC                = 00000014
SYS$DCLEXH                        ********   GX   05        UETUNT$V_TESTABLE              = 00000001
SYS$EXIT                          ********   GX   05        UETUNT$W_SIZE                  = 00000009
SYS$EXPREG                        ********   GX   05        UNIT_DESC                        000001ED R      02
SYS$FAO                           ********    X   05        UNIT_LIST                        00000A88 R      03
SYS$GET                           ********   GX   05        UNIT_LOOP                        00000C98 R      05
SYS$GETDEV                        ********   GX   05        UNIT_NUMBER                      000002E2 R      03
SYS$GETDVI                        ********   GX   05        UPDATE_FAILED                    00000CFD R      05
SYS$GETMSG                        ********   GX   05        WRITE_PRM                        000002FD R      02
SYS$INPUT                         00000061 R      02        WRITE_SIZE                     = 00000000
SYS$MGBLSC                        ********   GX   05        XD_CHAN                          00000306 R      03
SYS$OPEN                          ********   GX   05        XD_IOSB                          00000670 R      03
SYS$PUTMSG                        ********   GX   05        XM$M_CHR_LOOPB                 = 00000002
SYS$QIO                           ********   GX   05        XMIT                             00000556 R      05
SYS$QIOW                          ********   GX   05        XMIT_BUF                         00000680 R      03
SYS$SETAST                        ********   GX   05        XMIT_EFN                       = 00000005
SYS$SETIMR                        ********   GX   05
SYS$SETPRN                        ********   GX   05
SYS$SETSFM                        ********   GX   05
SYS$TRNLOG                        ********   GX   05
SYS$UPDATE                        ********   GX   05
SYS$WAITFR                        ********   GX   05
SYSIN_FAB                         00000A98 R      03
SYSIN_RAB                         00000AEB R      03
TEST_NAME                         0000000F R      02
TEST_OVERM                      = 00000002
TEST_OVERV                      = 00000001
TEXT_BUFFER                     = 000000FA
THREEMIN                          000001DD R      02
TIME_ERR_OUT                      0000099E R      05
TIME_ID_T                       = 00000001
TIME_SUC_OUT                      000009C9 R      05
TR_P1BUF                          0000038E R      03
TR_P2BUF                          000003B2 R      03
```

```
                                        +------------------+
                                        ! Psect synopsis !
                                        +------------------+

PSECT name                      Allocation              PSECT No.  Attributes
----------                      ----------              ---------  ----------
.  ABS  .                       00000000  (      0.)    00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                           00000000  (      0.)    01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
RODATA                          000005DC  (   1500.)    02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD   NOWRT NOVEC PAGE
RWDATA                          00000CAC  (   3244.)    03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD     WRT NOVEC PAGE
$RMSNAM                         00000023  (     35.)    04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
DMPF                            00000D51  (   3409.)    05 (  5.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE

                                     +------------------------+
                                     ! Performance indicators !
                                     +------------------------+

Phase                     Page faults    CPU Time         Elapsed Time
-----                     -----------    --------         ------------
Initialization                     40    00:00:00.07      00:00:00.58
Command processing                141    00:00:00.67      00:00:05.85
Pass 1                           1114    00:00:30.70      00:01:12.41
Symbol table sort                   9    00:00:03.39      00:00:07.81
Pass 2                            472    00:00:07.37      00:00:16.21
Symbol table output                39    00:00:00.30      00:00:01.08
Psect synopsis output               4    00:00:00.04      00:00:00.06
Cross-reference output              0    00:00:00.00      00:00:00.00
Assembler run totals             1822    00:00:42.55      00:01:44.02

The working set limit was 2000 pages.
171259 bytes (335 pages) of virtual memory were used to buffer the intermediate code.
There were 130 pages of symbol table space allocated to hold 2362 non-local and 39 local symbols.
1811 source lines were read in Pass 1, producing 41 object records in Pass 2.
59 pages of virtual memory were used to define 52 macros.

                                    +----------------------------+
                                    ! Macro library statistics !
                                    +----------------------------+

Macro library name                         Macros defined
------------------                         --------------
_$255$DUA28:[UETP.OBJ]UETP.MLB;1                        2
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                          1
_$255$DUA28:[SYSLIB]STARLET.MLB;2                      46
TOTALS (all libraries)                                 49

2653 GETS were required to define 49 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:UETDMPF00/OBJ=OBJ$:UETDMPF00 MSRC$:UETDMPF00/UPDATE=(ENH$:UETDMPF00)+EXECML$/LIB+LIB$:UETP/LIB
```

0410 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SATSSF18
LIS

SATSSS09
LIS

UETCOMS00
LIS

UETDISK00
LIS

SATSSS10
LIS

UETDMPF00
LIS

0411 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

UETFORT03
LIS

UETFORT01
LIS

UETLPAK00
LIS

UETNETS00
LIS

UETDRIW00
LIS

UETFORT02
LIS